DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: AC Print, Octal Number, Sign, Single Column			LSR # OT 1.1 t
			Classification Closed
No. of Regs. in Subroutine	Temp. Regs. used by Subroutine	Average Time (operations)	Max. Time (operations)
35	1t, 2t	61.5	63.0

Preset Parameters

none

Program Parameters
on entering Subroutine

ac: Number to be printed

Results
on leaving Subroutine

ac: 0

Description

When control is transferred to this routine, the number to be printed should be in the accumulator. A five digit octal number is printed preceded by either a "0" for a positive or a "1" for a negative number. After the number is printed a carriage return is executed so that continual use of this routine will result in a single column of numbers.

Notes:

1. No point is printed.
2. The qp orders are set to suppress the punch.

MD, Nov.9,1951
JWC III, Nov.13,1951

DL-291

TITLE: AC Print, Octal Number, Sign, Single Column LSR # OT 1.1 t

Abstract: This subroutine prints a five digit octal number with a "0" or "1" and no point preceding it and then a carriage return. A "0" designates a positive number and a "1" a negative number.

Upon entering the subroutine:

AC: number to be printed

Temporary Registers:

1t: register used to store the remainder of the word.
2t: register used to store the digit counter.

```
       00   ta 19r    Set return address
       01   ts 1t     Store value
       02   cp 20r    Is word negative?
       03   ca 26r    No. Print "0"
23r → 04   |qp 144    Printing
       05   cs 25r  } Set digit
       06   ts 2t   } counter
16r → 07   ca 1t      Value in AC
       08   sr*12
       09   ad 34r  } Add start of
       10   td 13r  } number table
       11   sl 15   } Store
       12   ts 1t   } remainder
       13   (ca 0)  } Put flexo code for
                    } digit in AC
       14   qp 128    Print digit
       15   ao 2t   } Have all digits been
       16   cp 7r   } printed? No.
       17   ca 4r   } Yes. Cause a
       18   qp 128  } carriage return
       19   (sp 0)  } Go back to main
                    } program
 2r → 20   ad 24r     Change sign
       21   ts 1t     Store value
       22   ca 27r    Print "1"
       23   sp 4r
       24   0.77777
       25   p4        Initial value of counter
       26   p45     }
       27   p36     }
       28   p39     }
       29   p3      } Number
       30   p21     } Table
       31   p33     }
       32   p43     }
       33   p15     }
       34   p26r    }
```

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: AC Print and/or Punch, Octal Number (Magnitude), Sign, No Carriage Return			LSR # OT 1.2 t Classification Closed
No. of Regs. 33	Temp. Regs. 3	Average Time (operations) 7 print	Max. Time (operations) 7print

Preset Parameters

v1 p0 (does not need to be inserted) to print, or p64 to punch, or p128 to punch and print simultaneously

Program Parameters

ac: Number to be recorded

Results

ac: Number has been recorded and is no longer available, C(AC) = p1

Description

When control is transferred to this subroutine, the number to be printed should be in the accumulator. A five digit octal number is printed preceded by either a "+" for a positive number or a "-" for a negative number, i.e. 0.12345→+12345 and 1.12345→-65432.

Notes:

1. No point is printed
2. No carriage return or space is printed
3. The qp instructions are normally set to print and suppress the punch.

TITLE: AC PRINT AND/OR PUNCH, OCTAL NUMBER (MAGNITUDE), SIGN, NO CARRIAGE RETURN — LSR# OT 1.2 t

Abstract: This subroutine records a five digit octal number with a "+" or a "-" and no point preceding it. An initial "+" designates a positive number and an initial "-" a negative number; i.e. 0.12345→+12345 and 1.12345→-65432 respectively.

Preset Parameter:
vl: p0 to print (does not need to be inserted), or p64 to punch, or p128 to punch and print simultaneously.

Upon entering the subroutine:
AC: number to be recorded

Temporary Registers:
d: unused
1t: register used to store the remainder of the word.
2t: register used to store the digit counter.

```
Enter→00   ta 17r    Set return address
      01   ts 1t     Store number
      02   cp 18r    Is number positive?
      03   ca 23r    Put Flexo code for "+" in AC
21r--→04   qp128sl   Print sign
      05   sr *12  } Set digit
      06   ts 2t   } Counter
16r--→07   ca 1t     Value in AC
      08   sr *12
      09   ad 22r  } Add start of
      10   td 13r  } number table
      11   sl 15   } Store
      12   ts 1t   } Remainder
 (10r)13   ca (0)  } Put Flexo code for
                   } digit in AC
      14   qp128sl   Print digit
      15   ao 2t   } Have all digits
      16   cp 7r   } been printed?
  (0r)17   sp (0)    Return to main program
2r--→18    cm 1t     Positive magnitude in AC
      19   ts 1t     Store number
      20   ca 24r  } Put Flexo code for
      21   sp 4r   } "-" in AC
      22   p25r      Start of number table
      23   1.30043   "+" character
      24   1.30007   "-" character
      25   p45     }
      26   p36     }
      27   p39     }
      28   p3      }  Number
      29   p21     }  Table
      30   p33     }
      31   p43     }
      32   p15     }
```

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Print C(AC) as Octal Number, Sign Digit and Complement, Point, Single Column Layout.	LSR OT 1.3t TAPE T542-3 Classification Closed

No. of Regs. in Subroutine	Temp. Regs. used by Sub.	Average Time (operations)	Max. Time (operations)
36	3	8 print	8 print

Preset Parameters

v1 P0 (does not need to be inserted) to print, or P64 to punch, or P128 to punch and print simultaneously.

v2 (desired digit length) - 5

Program Parameters

on entering Subroutine

ac: number to be recorded

Results

on leaving Subroutine

ac: number has been recorded and is no longer available, C(AC) = qp 144 sl.

Description

When control is transferred to this subroutine, the number in the accumulator is printed as a five digit octal number preceded by either a "0" for a positive or a "1" for the complement of a negative number and an octal point. After the last digit is printed a carriage return is executed so that a single column of numbers is tabulated whenever the subroutine is used repeatedly.

Notes:

1. The qp instructions are normally set to print and suppress the punch. By inserting p64 or p128 in preset parameter v1, one can punch and suppress the printer or punch and print simultaneously.

2. Form of output is

0.12345 appears as 0.12345 and

1.12345 appears as 1.12345.

MSD	AS
1/23/52	2/5/52

TITLE: PRINT C(AC) AS OCTAL NUMBER, SIGN DIGIT AND COMPLEMENT, POINT, SINGLE COLUMN LAYOUT

LSR - OT 1.3t
TAPE T 542-3

Abstract: This subroutine prints a five digit octal number with a "0" or "1" and a point preceding it and then a carriage return. An initial "0" designates a positive number and an initial "1" the complement of a negative number.

Preset Parameters:
- v1: p0 (does not need to be inserted) to print, or p64 to punch, or p128 to punch and print simultaneously.
- v2: (desired digit length) - 5.

Upon entering the subroutine:
- AC: number to be recorded

Temporary registers:
- d: unused
- 1t: register used to store the remainder of the word.
- 2t: register used to store the digit counter.

```
        00  ta 20r     Set return address
        01  ts 1t      Store number
        02  cp 21r     Is number positive?
        03  ca 27r     { Put Flexo code for "0"
                       { and point in AC
24r --> 04  qp 134sl   Print{"0"/"1"}, shift
        05  qp 128sl   Print point
        06  cs 35r  }  Set digit counter
        07  ts 2t   }
17r --> 08  ca 1t      Number in AC
        09  sr* 12
        10  ad 26r  }  Add start of number
        11  td 14r  }     table
        12  sl 15   }  Store remainder
        13  ts 1t   }
(11r)-> 14  ca (0)     { Put Flexo code for
                       { digit in AC
        15  |qp 144sl  Print digit
        16  ao 2t   }  Have all digits
        17  cp 8r   }   been printed?

        18  ca 15r    }  Print a carriage
        19  qp 128sl  }   return
  (0r)  20  sp 0         Return to main program
  2r -> 21  ad 25r       Change sign
        22  ts 1t        Store number
        23  ca 28 r      { Put flexo code for "1"
                         { and point in AC
        24  sp 4r
        25  0.77777
        26  p27r         Start of number table
        27  0.07155   }    "." and "0"
        28  0.07144   }    "." and "1"
        29  p39       }
        30  p3        }
        31  p21       }  Number table
        32  p33       }
        33  p43       }
        34  p15       }
        35  p4a2         Value for digit
                           counter
```

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Print C(AC) as Decimal Fraction, Sign and Magnitude, Point, Single Column Layout.			LSR OT 2.2t Tape T664-3 Classification Closed
No. of Regs. in Subroutine 38	Temp. Regs. used by Sub. d - 2t (3)	Average Time (operations) 8 print	Max. Time (operations) 8 print

Preset Parameters

v1 p0 (does not need to be inserted) to print, or p64 to punch, or p128 to punch and print simultaneously

v2 (Desired digit length) - 5

Program Parameters

on entering Subroutine

ac: x, number to be recorded

Results

on leaving Subroutine

ac: number has been recorded and is no longer available, c(ac) = +0

Description:

When control is transferred to this subroutine the number in the accumulator is printed as a five-digit decimal fraction preceded by a sign and decimal point and followed by a carriage return. The last printed digit does not contain roundoff of additional digits.

Note:

1. The qp orders are normally set to print and suppress the punch. By inserting p64 or p128 in preset parameter v1 one can punch and suppress the printer or punch and print simultaneously.

MD	JWCIII	AS
1/18/52	1/21/51	1/23/52

TITLE: Print C(AC) as Decimal Fraction, Sign and Magnitude, Point, Single Column Layout.

LSR# OT 2.2t
Tape T664-3

Abstract: This subroutine prints a five digit decimal fraction preceded by a sign and decimal point and followed by a carriage return; i.e. 0.77776 appears as +.99993 and 1.77776 as -.00006. The last printed digit does not contain roundoff of additional digits.

Preset Parameters:

v1: p0 to print, or p64 to punch, or p128 to print and punch simultaneously.

v2: (Desired digit length) - 5.

Upon entering the subroutine:

AC: x, number to be recorded

Temporary Registers:

d: unused

1t: register used to store remainder of the word.

2t: register used to store the digit counter.

	00	ta 20r	Set return address
	01	ts 1t	Store number
	02	cp 21r	Is number positive?
	03	ca 25r	Put Flexo code for "+" and dec. point in AC
22r→	04	qp 134 sl	Print {±}, shift
	05	qp 128 sl	Print decimal point
	06	cs 37r	Set digit counter
	07	ts 2t	
17r→	08	cm 1t	|x| in AC
	09	mh 23r	$\|x\| \cdot 10 \times 2^{-15}$
	10	ad 24r	Add start of number table
	11	td 14r	
	12	sl 15	Store remainder
	13	ts 1t	
(11r)	14	ca(0)	Flexo code for digit in AC
	15	qp 128 sl	Print digit
	16	ao 2t	Have all digits been printed?
	17	cp 8r	
	18	ca 19r	Print a carriage return
	19	qp 144 sl	
(0r)	20	sp(0)	Return to main program
2r→	21	ca 26r	Put Flexo code for "-" and dec. point in AC
	22	sp 4r	Return to print
	23	p 10	10×2^{-15}
	24	p 27r	Start of number table.
	25	0.07143	+. character
	26	0.07107	-. character
	27	p 45	Number table
	28	p 36	
	29	p 39	
	30	p 3	
	31	p 21	
	32	p 33	
	33	p 43	
	34	p 15	
	35	p 13	
	36	p 49	
	37	p 4 a2	Value for digit counter

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Print Magnitude as Decimal Integer from AC, Initial Zero Suppression, No Carriage Return			LSR# OT 2.5 t Classification Closed
No. of Regs. in Subroutine 45	Temp. Regs. used by Sub. 4	Average Time (operations) 5 print	Max. Time (operations) 5 print

Preset Parameter
v1 p0 (does not need to be inserted) to print, or p64 to punch, or p128 to punch and print simultaneously

Program Parameters
on entering Subroutine
AC: integer to be printed

Results
on leaving Subroutine
Number printed out on typewriter

Description

This subroutine prints out the magnitude of the number contained in the accumulator as a decimal integer. If $x \cdot 2^{-15}$ is the binary number in the accumulator, the decimal digits of the equivalent decimal integers d_i are obtained as follows:

$$d_1 = \left[(x \cdot 2^{-15})\left(\frac{2^{11}}{10^4}\right)2^{-11}\right] = \left[\frac{x}{10^4}\right] \cdot 2^{-15}$$

where [ ] = "integral part of"

$$d_2 = \left[\left(\frac{x \cdot 2^{-15}}{10^4} - \left[\frac{x}{10^4}\right]10^4 \cdot 2^{-15}\right)(2^4)\left(\frac{10}{2^4}\right)\right]$$

$$= \left[\frac{x - \left[\frac{x}{10^4}\right] \cdot 10^4}{10^3}\right] \times 2^{-15}$$

etc.

This form is used so that none of the manipulations will yield an overflow. Actually, because the number $2^{11}/10^4$ is not

TITLE: Print Magnitude as Decimal Integer from AC, Initial Zero Suppression, No Carriage Return	LSR# OT 2.5 t

exact in the machine, a binary round-off is added at one place in the program to give a correct answer.

Notes:

1. Initial zeroes are suppressed. Zero will be printed as five spaces.
2. The form of output is :

```
   63
17352
    1
        (zero)
   27
```

3. The program resets automatically.
4. There is no carriage return.

TITLE: Print Magnitude as Decimal Integer from AC, Initial Zero Suppression, No Carriage Return — LSR# OT 2.5 t

Abstract: This subroutine prints out the magnitude of the number contained in the AC as a decimal integer. If $x \cdot 2^{-15}$ is the binary number in the AC, the decimal digits of the equivalent decimal integers d_i are obtained as follows:

$$d_1 = \left[(x \cdot 2^{-15})(2^{11}/10^4)(2^{-11})\right] = \left[\frac{x}{10^4}\right] \times 2^{-15}$$

where $[\ \ ]$ = "integral part of"

$$d_2 = \left[\left(\frac{x \cdot 2^{-15}}{10^4} - \left[\frac{x}{10^4}\right]10^4 \cdot 2^{-15}\right)(2^4)\left(\frac{10}{2^4}\right)\right] = \left[\frac{x - \left[\frac{x}{10^4}\right]10^4}{10^3}\right] \times 2^{-15}$$

etc.

This form is used so that none of the manipulations will yield an overflow. Actually, because the number $2^{11}/10^4$ is not exact in the machine, a binary round-off is added at one place in the program to give a correct answer.

Preset Parameters

vl p0 (does not need to be inserted) to print, or p64 to punch, or p128 to punch and print simultaneously

Temporary Registers

- d unused
- 1t temporary storage
- 2t digit counter
- 3t suppressor

Enter→00	ta 14r	Set return address
01	ts 21r	Store number in AC
02	cs 31r	Set digit Counter
03	ts 2t	
04	ca 0	Set zero Suppressor
05	ts 3t	
06	cm 21r	$\times 2^{11}/10^4$
07	mh 32r	
27r→08	sr *11	$\times 2^{-11}$ and store
09	ts 21r	
10	sl *15	Subtract off Integral part
11	ts 1t	

TITLE: Print Magnitude as Decimal Integer from AC, Initial Zero Suppression, No Carriage Return

LSR# OT 2.5 t

	12	ao 2t	Advance Counter	
	13	cp 15r		
(0r)	14	sp (0)	Return to main program	
13r→	15	ca 21r	Check for Initial Zero	
	16	su 3t		
	17	cp 28r		
	18	ca 21r	No initial zero; Add in table entry	
	19	ad 33r		
	20	ts 21r	Set Printer	
(1r)(20r)	21	(p0)		
	22	qp128sl	Print	
	23	cs 0	Eliminate Zero Suppressor	
	24	ts 3t		
30r--→	25	ao 1t	Round-off	
	26	mh 34r	Multiply by 10/16	
	27	sp 8r	Recycle	

17r→	28	ca 29r	Print Space
	29	qp136sl	
	30	sp 25r	
	31	p5	Counter
	32	0.15067	$\approx 2^{11}/10^4$
	33	ca 35r	Table entry
	34	0.50000	10/16
	35	p45	Table of Decimal Flexowriter Characters
	36	p36	
	37	p39	
	38	p3	
	39	p21	
	40	p33	
	41	p43	
	42	p15	
	43	p13	
	44	p49	

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Print C(AC) as Decimal Integer, Magnitude only, Initial Zero Suppression, Print Final Zero, No Layout	LSR OT 2.51t TAPE T-883 Classification Closed

No. of Regs. in Subroutine	Temp. Regs. used by Sub.	Average Time (operations)	Max. Time (operations)
49	4 (d - 3t)	5 print	5 print

Preset Parameters

vl p0 (need not be inserted) to print, or p64 to punch, or p128 to punch and print simultaneously

Program Parameters

on entering Subroutine

AC: Integer to be printed

Results

on leaving Subroutine

Number printed out on typewriter

Notes:

1. Initial zeroes are suppressed. Zero will be printed out as four spaces and a zero.

2. The form of output is:

```
    63
 17352
     1
→→→→0 (zero)
    27
```

3. The program resets automatically.

4. There is no carriage return.

JWC III	MAS
2/10/52	2/11/52

TITLE: Print C(AC) as Decimal Integer, Magnitude only, Initial Zero Suppression, Print Final Zero, No Layout — LSR CT 2,51t — TAPE T-883

Abstract: This subroutine prints out the magnitude of the number contained in the accumulator as a decimal integer. If $x \cdot 2^{-15}$ is the binary number in the accumulator, the decimal digits of the equivalent decimal integers d_i are obtained as follows:

$$d_1 = \left[\left(x \cdot 2^{-15}\right) \left(\frac{2^{11}}{10^4}\right)\left(2^{-11}\right) \right] = \left[\frac{x}{10^4}\right] \cdot 2^{-15}$$

where $[\;]$ = "integral part of"

$$d_2 = \left[\left(\frac{x \cdot 2^{-15}}{10^4} - \left[\frac{x}{10^4}\right] 10^4 \cdot 2^{-15} \right) \left(2^4\right)\left(\frac{10}{2^4}\right) \right]$$

$$= \left[\frac{x - \left[\frac{x}{10^4}\right] \cdot 10^4}{10^3} \right] \times 2^{-15}$$

This form is used so that none of the manipulations will yield an overflow. Actually, because the number $2^{11}/10^4$ is not exact in the machine, a binary round-off is added at one place in the program to give a correct answer.

Temporary Registers

- d - unused
- 1t - temporary storage
- 2t - digit counter
- 3t - suppressor

Preset Parameters

- vl — p0 (need not be inserted) to print, p64 to punch, or p128 to print and punch

	00	ta 19r	Set return address
	01	ts 25r	Set no. in AC
	02	cs 35r	Set digit counter
	03	ts 2t	
	04	ca 0	Set zero suppressor
	05	ts 3t	
	06	cm 25r	$\times \frac{2^{11}}{10^4}$
	07	mh 36r	
31r →	08	sr* 11	$\times 2^{-11}$
	09	ts 25r	
	10	sl* 15	Subtract off integral part
	11	ts 1t	
	12	ao 2t	Advance counter
	13	cp 20r	Are we at last digit?
	14	ca 25r	Last digit
	15	ad 37r	Add in start of table
	16	ts 17r	
(16r)	17	(p0)	Print
	18	qp 128sl	
(0r)	19	sp ( )	Return to main program
13r →	20	ca 25r	Not at last digit
	21	su 3t	Check for initial zero.
	22	cp 32r	
	23	ad 37r	Add in table entry
	24	ts 25r	
(24r)	25	(p0)	Set Printer
	26	qp 128sl	Print
	27	cs 0	Eliminate

TITLE: Print C(AC) as Decimal Integer, Magnitude only, Initial Zero Suppression, Print Final Zero, No Layout

LSR OT 2.51t
TAPE T-883

```
         28  ts 3t      zero suppressor       39  p45  ⎫
34r --> 29  ao 1t      Round off              40  p36  ⎪
         30  mh 38r                            41  p39  ⎪
         31  sp 8r                             42  p3   ⎪
22r --> 32  ca 33r  ⎫                          43  p21  ⎬ Table of Digits
         33  qp 136sl ⎬ Print space            44  p33  ⎪
         34  sp 29r  ⎭                         45  p43  ⎪
         35  p4                                46  p15  ⎪
         36  0.15067  2^11/10^4                47  p13  ⎪
         37 || ca 39r                          48  p49  ⎭
         38  0.50000  10/16
```

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Print Out Flexowriter Characters, Previously Stored Three Characters to a Register by LSR OT 3.lat, Using Program Parameter	LSR OT 3.lt TAPE T782-2 Classification Closed

No. of Regs. in Subroutine	Temp. Regs. used by Sub.	Average Time (operations)	Max. Time (operations)
38	d - 4t (5)	13 + 14n (n=no. of chars.) (Actually limited by number of printouts)	13+20n (n=no. of chars.) (Actually limited by no. of printouts)

Preset Parameters

vl: p0 (does not need to be inserted) to print, p64 to punch, and p128 to print and punch simultaneously.

Program Parameters

on entering Subroutine

ul: p(address of first register in storage in which characters have been previously stored.)

Results

on leaving Subroutine

Flexowriter characters printed out from storage.

Description

This subroutine prints out Flexowriter characters previously stored by LSR OT 3.lat, three to a register. Subroutine OT 3.lat stores characters 5-digits long, with a 00000 "tag" character indicating when the final digit is to be a 0 and when a 1. (For further description of LSR OT 3.lat, see the Specification sheet for LSR OT 3.lat). This subroutine reverses the process, unstacking and printing out. It will print out characters, placing a zero in the sixth digit, until a "tag" character is reached. Then it will place 1's in the sixth digit until another "tag" character is reached, etc. This is the exact reverse of LSR OT 3.lat.

When a negative number is reached (previously stored by the read-in routine) the routine stops printing and returns control to the main program.

Note

For notes on how many characters may be stored, etc., see the Specification Sheet for LSR OT 3.lat.

JWC III	AS
2/1/52	2/5/52

TITLE: Print Out Flexowriter Characters, Previously Stored Three Characters to a Register by LSR OT 3.lat, Using Program Parameter

LSR# OT 3.lt
TAPE# T782-2

Upon entering the subroutine:

sp (subroutine)
p (first address to be printed out of)

Preset Parameters:

vl: p0 (does not need to be inserted) to print, or p64 to punch, or pl28 to print and punch simultaneously

Temporary Registers:

d: temp. character storage
lt: 3 counter
2t: operational register
3t: find digit
4t: digit to be exchanged

	00	ta lr	Plant ul address
(3r)(0r)	01	ca (0)	Plant first address
	02	td llr	
	03	ao lr	Set return address
	04	td 13r	
	05	ca 0	Set final digit
	06	ts 3t	
	07	ca 36r	Set digit to be exchanged to pl
	08	ts 4t	
28r →	09	cs 37r	Set 3-counter
	10	ts lt	
(2r)	11	ca (0)	Pick up register to be printed
	12	ts 2t	
(4r)	13	cp (0)	Are we finished?
26r →	14	ca 2t	No. Shift character into position
	15	sr* 10	
	16	su 0	Is this character a zero?
	17	cp 29r	
	18	ts d	No
	19	sl* 15	Shift off left and restore reg. contents
	20	ts 2t	
	21	ca d	Add in final digit
	22	sl* 1	
	23	ad 3t	
	24	qp 128sl	Print
35r →	25	ao lt	Are we finished with registers?
	26	cp 14r	
	27	ao llr	Advance register no. and recycle
	28	sp 9r	
17r →	29	ca 2t	Restore register contents shifted.
	30	sl* 5	
	31	ts 2t	
	32	ca 4t	Exchange final digit
	33	ex 3t	
	34	ts 4t	
	35	sp 25r	Back to check finish
	36	pl	final digit
	37	p2	character counter

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Read in Standard Flexowriter Characters and Store Three Characters to a Register, Using Program Parameter. (For Use with LSR OT 3.1t.)			LSR OT 3.1at TAPE T781-2 Classification Closed
No. of Regs. in Subroutine 66	Temp. Regs. used by Sub. d - 6t (7)	Average Time (operations) 40 + 13n, where n is no. of characters (Present limiting case is no. of read-ins on PETR.)	Max. Time (operations) 40 + 26n (in the most unusual case) (Present limiting case is no. of read-ins on PETR)

Program Parameters
on entering Subroutine
ar: u_1 (automatic)
~~ul:~~ p (address of first register in which characters are to be stored)

Results
on leaving Subroutine

Flexowriter characters stored, five binary digits to the character, beginning at the register given in ul, three characters to the register. All registers are positive except the one succeeding the final storage register, which is set to n0.

Description

This subroutine reads in Flexowriter characters continuously from the photoelectric reader, and stores them in the registers beginning at the address stored in ul. The standard Flexowriter tape to be called in is typed on a Flexowriter having a binary code the same as the code on the Flexowriter connected to the Whirlwind output.

A "stop" character, with a seventh hole, must be typed at the end of the tape as a "tag" character so that the subroutine may stop the storage process.

Notes

1. The Flexowriter characters are shortened to five-digit length, and stored three to a register. A second "tag" character, 00000, is used to indicate whether or not the sixth digit in a character is to be a zero or one. Whenever a series of characters switches from final digit "zero", to final digit "one", and vice versa, such a tag is inserted. This is not necessarily a waste of storage, since all alphabetical characters in the standard Flexowriter code have the sixth hole zero, and thus there is no need for such a tag in most storage of straight language. Similarly, all numbers and most punctuation marks have the sixth hole a "one", and so a series of numbers does not require such a "tag" character until an alphabetical character appears. Mixtures of constantly changing letters and

TITLE: Read in Standard Flexowriter Characters and Store Three Characters to a Register, Using Program Parameter. (For Use with LSR OT 3.1t.)

LSR OT 3.1at

TAPE T781-2

numbers will, however, be wasteful of storage; and in such rare cases, another pair of storage and print routines should be used. With ordinary typescript, an estimated 25% of storage can be gained, however, over "two-to-a-register" methods of storing Flexowriter standard characters.

2. The number of registers needed to store a given tape will thus be a function of the number of changes from "sixth-hole-zero" to "sixth-hole-a-one". In standard English language, an average formula would seem to be

$$N = \frac{n}{3} + \frac{n}{20} = \frac{23n}{60} = .38n$$

Where N is the number of registers needed for storage, and n the number of characters, n/20 being considered an average number of such changeovers.

3. This subroutine must always be used in conjunction with Print-Out Routine OT 3.1t. It may be used in the following manner: Read in subroutine OT 3.1t, preferably in the upper or lower ends of storage. When control is transferred to this subroutine it will automatically read in and store the required standard Flexowriter characters in the addresses beginning at the address given by program parameter ul. After the final stop character is reached, it will not store this, but instead make the succeeding register a negative zero. Control is then automatically transferred to the instruction following the ul register.

Following this, OT 3.1t may be read in by means of the standard read-in conversion program, or else, if it has been previously stored, control may be transferred to it at any desired time. For a fuller explanation of how this program prints-out, see the Specification Sheet for OT 3.1t.

4. The character 000001 cannot be read in by this subroutine, since it will be confused by the machine with the 5-digit "tag" character 00000. However, in the 1951 Flexowriter code, this is "back-space", which should not be needed, and in the 1952 Flexowriter code, this character is not produced by any key on the keyboard.

JWC III 1/24/52	AS 2/5/52

TITLE: Read in Standard Flexowriter Characters and Store Three Characters to a Register, Using Program Parameter. (For Use with LSR # OT 3.1t.)

LSR# OT 3.1at
TAPE# T781-2

Upon entering the subroutine:

- sp: subroutine
- p: first address to be stored

Temporary registers:

- d: temporary digit store
- 1t: last 6th digit representation
- 2t: character store
- 3t: present 6th digit representation
- 4t: temporary store
- 5t: 3-counter
- 6t: temporary store

```
              00  ta 1r      Plant ul address
(3r)(0r)      01  ca ( )     } Plant 1st address where
              02  td 53r     } machine is to store chars.
              03  ao 1r      } Set return address
              04  td 42r     }
              05  cs0        } Set last digit reg. in
              06  ts 1t      } for zero
              07  cs 64r     } Store 3 counters
              08  ts 5t      }
              09  ca0        } Set zero in temp
              10  ts 4t      }    store
29r -->       11  qr0        Read in
              12  ca 8       }
              13  sr* 5      }
              14  td d       } Store 6 digits
              15  ca d       }   in 2t
              16  sr* 5      }
              17  ts 2t      }
              18  su 65r     } Check to see if
              19  cp 30r     } char. is a"stop"
```

```
33r -->       20  ca 2t      }
              21  sl* 14     } Store present 6th
              22  sr* 14     } digit representation
              23  su0        }
              24  ts 3t      }
              25  mh 1t      } Are past and present
              26  cp 43r     } 6th digits the same?
47r -->       27  ca 2t      } Store 5 digits of
              28  sp 48r     }   character
              29  sp 11r     Recycle
19r -->       30  ca 65r     }
              31  su 2t      } Is char. a"stop"?
              32  cp 34r     }
              33  sp 20r     No. Back to check digits
32r -->       34  ca 0       Yes.
              35  sp 48r     Store 00000
              36  ca 0       } Store 00000
              37  sp 48r     }
              38  ca 53r     } Plant last register
              39  td 41r     }    address
```

TITLE: Read in Standard Flexowriter Characters and Store Three Characters to a Register, Using Program Parameter. (For Use with LSR # OT 3.1t.)

LSR# OT 3.1at
TAPE# T781-2

```
                 40  cs 0     } Put negative no.
39r ----->       41  ts (0)   }  in last reg.
(04r) --->       42  sp (0)     Back to main Program
26r ----->       43  ca 3t    } Switch  digit
                 44  ts 1t    }  representations
                 45  ca 0     } Store 00000
                 46  sp 48r   }
                 47  sp 27r     Back to store 5 digits
28r,37r } -->    48  ta 63r     Plant link of internal
35r,46r }                        subroutine
                 49  sr* 1    }
                 50  ts 6t    }
                 51  ca 6t    }
                 52  ad 4t    } Shift left and store
(62r)(02r)       53  ts (0)   }
                 54  sl* 5    }
                 55  ts 4t    }

                 56  ao 5t    } Are we at end
                 57  cp 63r   }  of register?
                 58  ca 0     } Yes. Reset temp.
                 59  ts 4t    }    store.
                 60  cs 64r     Reset 3-counter
                 61  ts 5t
                 62  ao 53r     Advance Storage
57r --> (43r)    63  sp (0)     Exit from internal s.r.
                 64  p2         3-counter reset
                 65  p51        "stop" character
```

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: (30,0,0) MRA Print and/or Punch, Decimal Fraction, Sign, Number of Digits Arbitrary, No Carriage Return, Sign Agreement (Interpreted)			LSR OT 103.10t TAPE T799-1 Classification Closed, Interpreted
No. of Regs. in Subroutine 76	Temp. Regs. used by Sub. 4	Average Time (operations) 94+19n, number of digits printed = n	Max. Time (operations) 94 + 27n

Preset Parameters

vx: pN, where N is the address in storage of the first register of the interpretive subroutine (in title of main program)

v1: p0 (does not need to be inserted) to print, p64 to punch, or p128 to print and punch simultaneously

v2: pn, where n is the number of decimal digits to be printed.

Description

This subroutine prints and/or punches the sign and magnitude of the contents of the MRA in the following manner

$$\pm .d_1 d_2 \ldots d_n$$

The number, n, of decimal digits to be printed is a preset parameter (v2). The digits, d_i, are obtained by multiplying the magnitude of the contents of the MRA successively by p10.

This subroutine contains a sign agreement program so that the contents of the MRA need not be a number whose major and minor parts are of like sign.

The sp instruction transferring control to this subroutine must be an interpreted sp (i.e., control must be in the interpretive subroutine). After execution of the subroutine control remains in the interpretive subroutine which then proceeds to interpret the instruction following the sp instruction in storage.

There is no carriage return.

This subroutine can be used with any (30,0,0) interpretive subroutine. The contents of the MRA are left undisturbed during the execution of this subroutine.

FCH	MRS
2/1/52	2/8/52

TITLE: MRA Print and/or Punch in (30,0,0) Interpretive Subroutine without Sign Agreement, Decimal Number, Sign, Number of **Digits Arbitrary, No Carriage Return (Interpreted)** — LSR OT 103.10t, TAPE T799-1

Abstract: This subroutine prints out a ± sign and a decimal point followed by the magnitude of the contents of the MRA as a decimal fraction. The decimal digits are obtained by multiplying the contents of the MRA successively by p10. The number of digits to be printed is a preset parameter (v2). There is no carriage return. The subroutine is interpreted and can be used with any (30,0,0) interpretive subroutine.

Preset Parameters

- vx: pN, where N is the address in storage of the first register of the interpretive subroutine (in title of main program)
- v1: p0 (does not need to be inserted) to print, or p64 to punch, or p128 to punch and print simultaneously
- v2: pn, where n is the number of decimal digits to be printed

Temporary Storage

- d unused
- 1t, 2t, 3t: Temporary storage
- 4t Digit counter

```
          00 ||ta 37r
          01 ||sp ax       Resume ordinary ww operation
          02  ca 3ax  ┐
          03  ts 2t   │    Store C(mra) in 1t and 2t
          04  ca 2ax  │
  58r --> 05  ts 1t   ┘
  54r --> 06  mr 2t        Do 1t and 2t agree in sign?
          07  cp 38r
          08  ca 1t   ┐
  56r --> 09  cp 12r  │
          10  ca 74r  │    Sense and print algebraic
          11  sp 13r  │      sign of C(mra) followed
   9r --> 12  ca 75r  │      by a decimal point.
  11r --> 13  qp 134sl│
          14  qp 128sl┘
          15  cs 61r  ┐    Set up digit counter
          16  ts 4t   ┘
  35r --> 17  cm 1t   ┐
          18  mh 62r  │
          19  ts 3t   │
          20  sl 15   │
          21  ts 1t   │
          22  cm 2t   │    Multiply C(1t,2t) by p10
          23  mh 62r  │
          24  ts 2t   │
          25  sl 15   │
          26  ex 2t   │
          27  sa 1t   │
          28  ts 1t   ┘
          29  ca 3t   ┐
          30  ad 63r  │    Set up entry into table
          31  td 32r  ┘
   (31r)  32  ca (0)  ┐    Print a single digit
          33  qp 128sl┘
```

TITLE: MRA Print and/or Punch in (30,0,0) Interpretive Subroutine without Sign Agreement, Decimal Number, Sign, Number of **Digits Arbitrary, No Carriage Return (Interpreted)**

LSR OT 103.10t
TAPE T799-1

```
             34  ao 4t    }  Have enough digits       48r --> 55  ao 1t
             35  cp 17r   }  been printed?                    56  sp 9r
             36  sp ax    }  Return control to        40r --> 57  cs 1t
      (0r)   37 ||sp (0)  }  int. subroutine                  58  sp 5r
7r -->       38  cm 1t    }                                   59 ||ca0
             39  su 0     }  Is C(1t) ≠ 0 ?                   60 ||p1
             40  cp 57r   }                                   61 ||n1a2        v2
             41  cm 2t    }                                   62 ||p10
             42  su 0     }  Is C(2t) ≠ 0?                    63 ||p64r
             43  cp 51r   }                                   64 ||p45         0      Table
             44  su 60r   }                                   65 ||p36         1
             45  ad 59r   }  Form 1 - |C(2t)|                 66 ||p39         2
             46  ts 2t    }                                   67 ||p3          3
             47  ca 1t    }  Is C(1t) pos.?                   68 ||p21         4
             48  cp 55r   }                                   69 ||p33         5
             49  su 60r   }  Form C(1t) - 2^-15               70 ||p43         6
             50  ts 1t    }                                   71 ||p15         7
43r -->      51  cs 2t    }  Complement C(2t)                 72 ||p13         8
             52  ts 2t    }                                   73 ||p49         9
             53  ca 1t    }  Re-enter sign agree-             74 ||0.07143     +.
             54  sp 6r    }  ment                             75 ||0.07107     -.
```

PROGRAMMED ARITHMETIC

PA 1.2 t (T747-3) is the same as PA 1.1 t except that register 26r reads pax2.

PA 2.2 t (T723-1) is the same as PA 2.1 t except that register 197r reads pax2.

Both the above subroutines use the new Flexowriter code whereas PA 1.1 t and PA 2.1 t both use the old Flexowriter code.

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Operations on Real (15,15,0) Floating Point Double Register Numbers (General Subroutine)			LSR PA 1.1 t Classification Interpretive
No. of Regs. in Subroutine 97	Temp. Regs. used by Sub. -----	Average Time (operations) see description	Max. Time (operations) see description

Preset Parameters (to be typed in the title of the main program)
v8/ pk: k = separation between two registers of the number
vx/ pN: N = location of first register of subroutine in storage

Description:

By means of this subroutine various logical and arithmetic operations can be carried out upon real numbers represented in the (15,15,0) system and stored in two registers, say n and n+k, where k is specified by a preset parameter.

If a is any number such that

$$2^{-15} < |a| < 2^{15}, \text{ or } a = 0,$$

then there exists a signed 15 binary digit number x and a signed 15 binary digit integer y such that

$$\tfrac{1}{2} \leq |x| < 1 \text{ or } x = 0,$$

$$0 \leq |y| < 2^{15},$$

and $$\left|\frac{a - x\cdot 2^{y}}{a}\right| \leq 2^{-16}.$$

x and y are stored respectively in the two registers n and n+k for use by this subroutine.

Example (a) Let a = -300 (decimal)
Then a = -100101100 (binary)
and hence x = 1.011010011111111
and y = 1001
Register n contains 1.011010011111111
and Register n+k contains 0.000000000001001

TITLE: Operations on Real (15,15,0) Floating Point Double Register Numbers (General Subroutine)	LSR PA 1.1 t

Example (b) Let $a = \frac{1}{10}$ (decimal)

Then $a = 0.0001100110011001100110011.....$
and hence $x = 0.110011001100110$,
and $y = -11$
Register n contains 0.110011001100110
and Register $\overline{n+k}$ contains 1.111111111111100

The programmer need not carry out this conversion process himself, but instead need only find a five decimal digit X and Y such that

$$\frac{1}{10} \leq | X | < 1, \text{ or } X = 0$$

$$0 \leq | Y | \leq 32,767$$

and $$\left| \frac{a - X \cdot 10^{Y}}{a} \right| \leq 10^{-5}$$

By giving this information to the proper subroutine in the IP section of the Library of Subroutines, the remainder of the conversion can be done automatically.

Operations upon numbers in this representation are coded using an instruction code similar to but differing somewhat from the standard WW code. Any number of these instructions can be performed in sequence by placing an sp ax before the first instruction of the sequence. The instructions are then interpreted successively until a change-of-control instruction is reached at which point another sequence is performed unless the instruction is an sp ax. In the latter case, ordinary WW operation is resumed at the register following the sp ax.

The subroutine contains the analogy of an accumulator and a program counter. The "program counter" is in register 76r and contains the address of the register containing the instruction being interpreted. The "multiple register accumulator" (mra) is in registers 2r,and 3r and contains the result of the last instruction (except ta, cp, and sp). Register 2r contains x and 3r contains y.

TITLE: Operations on Real (15,15,0) Floating Point Double Register Numbers (General Subroutine)	LSR PA 1.1 t

Instruction	Function
ts n	Store the number in the mra in registers n and n+k.
ta n	Store the address of the register following the last sp or cp instruction in the address section of register n.
ex n	Exchange the number in the mra with the number in registers n and n+k.
cp n	If the number in the mra is negative, proceed as in sp, otherwise ignore the instruction.
sp n	If n ≠ ax, interpret next the instruction in register n. If n = ax, resume ordinary WW operation at the register following the register containing the sp ax.
ca n	Clear the mra and then add into the mra the number in registers n and n+k.
cs n	Clear the mra and then add the negative of the number in registers n and n+k into the mra.
ad n	Add the number in registers n and n+k to the number in the mra and leave the result in the mra.
su n	Subtract the number in registers n and n+k from the number in the mra and leave the result in the mra.
cm n	Clear the mra and add the magnitude of the number in registers n and n+k into the mra.
mr n	Multiply the number in the mra by the number in registers n and n+k and leave the result in the mra.
dv n	Divide the number in the mra by the number in registers n and n+k and leave the result in the mra.

N.B. In the above the phrase "the number in" should be read to mean "the number represented by the contents of".

TITLE: Operations on Real (15,15,0) Floating Point Double Register Numbers (General Subroutine)	LSR PA 1.1 t

Notes:

1. Entering and Leaving the Subroutine
Wherever an sp ax is encountered, the machine will enter the subroutine if it is not already in it, or leave it if it is in the subroutine.

2. Preset Parameters
A general preset parameter, vx, is used to specify the location in storage of the first register of the subroutine. Another general preset parameter, v8, is used to specify the separation k of the two registers containing a number. Both of these parameters must be specified in the title of the program tape.

3. Accuracy of Arithmatic Operations
The operations multiply and divide give a result with at least 14 1/2 digit accuracy, i.e. the result is rounded off at worst in the 15th place. The relative error in the operations add and subtract is at worst 2^{-15}, but it may be biased downwards if the magnitude of the sum is sufficiently greater than the magnitude of either of the addends.

4. Alarms
Arithmetic overflow alarms can occur in registers 19r, 20r, 47r, 71r, or 72r. In all cases this occurs when the exponent y of the result being calculated during an ad, su, mr, or dv instruction lies outside the range (-2^{15}, 2^{15}). When such an alarm occurs, the address of the instruction that is being interpreted will be in register 76r. Any other alarms in the subroutine will be due to excessive addresses or use of an instruction not listed above.

TITLE: Operations on Real (15,15,0) Floating Point Double Register Numbers (General Subroutine) — LSN# PA 1.1 t

Instruction Code and Operation Times:

ts	23	cp	17(+), 27 (-)	cs	26	cm	25
ta	18	sp	24	ad	41	mr	23
ex	23	ca	27	su	43	dv	26

Preset Parameters: (to be typed in the title of the main program)

v8/ pk: k = separation in storage of two registers of numbers
vx/ pN: N = address in storage of initial register of subroutine

Entry	Loc.	Instruction	Comment
	00	ta 76r	Store address of 1st instruction to be interpreted
	01	sp 76r	
	02	(p0)	x_1 } multiple register
	03	(p0)	y_1 } accumulator
89r→	04	‖ca ax	"ca"
89r→	05	sp 8r	"cs"
89r→	06	sp 46r	"ad"
89r→	07	sp 44r	"su"
5r,89r→	08	ca 86r	"cm" Pick up and
	09	ts 93r	store cs, ca, cm } n+k
	10	sp 90r	
	11	p16	
89r→	12	mr 2r	"mr" Form $x_1 \cdot x_2$
	13	sp 69r	
89r→	14	ex 2r	"dv" Form and Store $\frac{x_1 \cdot 2^{-1}}{x_2} \cdot 2$ s.f.
	15	sr 1	
	16	dv 2r	
	17	sf 25r	
	18	ex 3r	
	19	su 95r	Form $y_1 - y_2 + 16$
	20	ad 11r	
	21	sp 72r	
33r→	22	cs 2r	Is $x_1 < 0$?
	23	cp 75r	
	24	sp 34r	Go to sp routine
	25	(p0)	digit storage
	26	pa8	separation parameter
89r→	27	sp 90r	"ts"
	28	‖sp 19r	
89r→	29	ca 42r	"ta" } Store digits in A-register
(78r)	30	ta (0)	
	31	sp 75r	
89r→	32	sp 90r	"ex"
89r→	33	sp 22r	"cp"
24r,89r→	34	ao 76r	"sp" } Set A-Register address
	35	td 42r	
	36	ca 88r	Set pick-up instruction's address
	37	td 76r	
	38	su 4r	Does address of cp or sp instruction differ from ax?
	39	ts 95r	
	40	cm 95r	
	41	su 0	
(35r)	42	cp (0)	
	43	sp 76r	
7r→	44	ts 91r	Complement x_2
	45	cs 91r	
6r→	46	ex 95r	Form and store $y_2 - y_1$.
	47	su 3r	
	48	ts 91r	
	49	cp 55r	Is $y_2 - y_1 > 0$?

TITLE: Operations on Real (15,15,0) Floating Point Double Register Numbers (General Subroutine)   LSR# PA 1.1 t

```
       50  ad 3r  )
       51  ts 3r  |  Interchange
       52  ca 2r  }  (x1,y1) and
       53  ex 95r |  (x2,y2)
       54  ts 2r  )
 49r-->55  cm 91r }  Store |y2-y1|
       56  td 62r }
       57  su 96r }  Is |y2-y1| > 15?
       58  cp 60r }
       59  sp 75r    Addition unnecessary
 58r-->60  ca 96r }  Set y2 = 15
       61  ex 95r }
  (56r)62  sr (0)    Form x2 2^-(y2-y1)
       63  sa 2r  }  Add
       64  ts 2r  }
       65  ca 0   )
       66  ex 2r  |  Add overflow into
       67  sr *15 }  2^-15 times the
       68  ad 2r  )  sum
 13r-->69  sf 25r }  Scale factor
       70  ex 3r  }  and store result
       71  ad 96r    Form y2+y1
 21r-->72  su 25r    Subtract scale factor
```

```
               94r-->73  ex 3r  }  Store result in
                     74  ts 2r  }  proper registers
         23r,31r,5r-->75  ao 76r }  Increase address of pick-up
                                   instruction
  (37r)(0r)1r,43r-->76  ca (0)    Pick-up instruction
                     77  ts 91r )
                     78  td 30r |  Store instruction
                     79  td 88r |  and digits
                     80  ad 26r |  where
                     81  ts 93r |  necessary
                     82  td 86r )
                     83  sr *11 )  Form sp to proper
                     84  ad 28r }  part of subroutine
                     85  td 89r )
                (82r)86  ca (0) }  Pick up and
                     87  ts 95r }  store y2
                (79r)88  ca (0)    Pick up x2
                (85r)89  sp (0) }  Go to part of subroutine for
                                   particular instruction
          10r,32r-->90  ca 2r
                (77r)91  (p0)
                     92  ex 3r
           (9r),(81r)93  (p0)
                     94  sp 73r
                     95  (p0)      Temporary storage
                     96  p15
```

4r-ca	14r-dv
5r-cs	27r-ts
6r-ad	29r-ta
7r-su	32r-ex
8r-cm	33r-cp
12r-mr	34r-sp

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Specifications of WHIRLWIND I LIBRARY SUBROUTINE Number PA 2.2

Title: Extra-Precision and Floating-Point Real Number Arithmetic, using 2-register 24,6,0 Numbers; Basic Instruction Code with Division, INTERPRETIVE

Total Number of Registers Occupied by the Subroutine:	204	storage registers
Temporary Storage Registers Required by the Subroutine:	no	temporary regs.
Time Required to Perform the Subroutine: average =	50*	WWI operations
maximum =	76*	WWI operations

*per interpreted operation; see page 4 for details

Preset Parameters (Values to be indicated in tape title line)

x	pN:	N = address assigned to the initial register of the subroutine
x2	pk:	k = separation between registers assigned to each 2-register number

---

Description

This interpretive subroutine, when called into action, takes instructions (more strictly, program parameters written as instructions) one at a time from consecutive storage registers and performs the designated single-address operations defined by the interpreted-instruction code given on page 4. These operations are primarily arithmetical operations performed on real numbers represented in the 24,6,0 system. Each number is stored in some multiple-register location n consisting of the pair of registers n and n+k, where n is the address of the given location and k is determined by preset parameter x2.

The 24,6,0 number system represents any real number N, provided that either N=0 or $2^{63} > |N| \geq 2^{-64}$, as a signed 24-binary-digit fraction x and a signed 6-binary-digit integer y, where x and y are chosen in such a way that either x=0 or $1 > |x| \geq .5$ and that $|1 - x2^{y}N^{-1}| < 2^{-24}$. Thus the number pair x,y represents N to within ±0.000006%, equivalent to about 7 significant decimal digits. The sign and first 15 digits of x occupy one register while the sign and 6 digits of y and the last 9 digits of x occupy the second register of the pair assigned to contain the number N. Details of this and other number systems are available elsewhere.

A multiple-register accumulator (MRA) is used in place of the AC in many interpreted operations. This MRA is not a special register as is the AC but rather is a group of 3 ordinary storage registers contained within the interpretive subroutine, specifically registers 2r, 3r, and 4r. Even though only 2 registers are needed to contain a 24,6,0 number, 3 registers are used for the MRA to avoid the time-consuming operation of packing the last 9 digits of the number and the sign and 6 digits of the exponent together into one register after each interpreted instruction. A further advantage is gained in that any sequence of arithmetic operations is performed using 30 digits for the number and 15 digits for the exponent. This provides in effect a 30,15,0 system. The 24 and 6 limitation is imposed only when necessary, namely on ts and ex operations. Thus greater range and greater precision are available in sequences of arithmetic operations than the 24,6,0 system would normally allow.

The roundoff error on ad and su is made in the 29th digit of the sum before it is scale-factored. That is, in adding any two 24,6,0 numbers, $v.2^w$ to $x.2^y$, assuming $1>|v|\geq .5$, $1>|x|\geq .5$, $w\geq y$, the sum obtained is $\left[(v + x.2^{y-w} \pm 2^{-29})\ 2^z\right]\ 2^{w-z} = u.2^{w-z}$, where z is chosen in such a way that $1>|u|\geq .5$.

The roundoff in mr is made in the 28th digit.

The roundoff in dv is made in the 27th digit.

The roundoff in ts and ex (i.e. in packing the 30,15,0 numbers into 24,6,0 form) is of course made in the 25th digit. If the exponent y is less than -63, the value -63 is substituted for it, without changing x in any way.

Arithmetic alarms, because of the floating point system employed, and because of the extended range allowed within the MRA, will normally not occur in an interpreted program unless the contents of the MRA, call it $x.2^y$, prior to a ts or ex operation has an exponent $y>63$, in which case an overflow alarm always occurs at register 203r during the interpretation of the ts or ex operation, even if x = 0. If during an arithmetic operation the exponent y exceeds the bounds $2^{15}>y>2^{-15}$, an overflow alarm will occur at register 28r, 85r, 130r, 175r or 176r.

Entry to and exit from the subroutine is accomplished by means of the instruction spax. The first instruction in a program is always performed in the Whirlwind code. When 24,6,0 operations are needed, control is transferred to the subroutine by spax, x being the parameter which specifies the location of the subroutine. Instructions following the first spax are then performed in the interpreted code. When operations on 1-register fixed-point words are desired, control is transferred back to the main program by spax. This spax is given a special interpretation by the subroutine and results in the instructions following it being performed in the Whirlwind code. Use of a sequence of Whirlwind-coded instructions between two interpreted instructions does not affect the contents of the MRA, but use of any interpreted instruction does affect the contents of the AC.

For numerical input at the present time, all decimal numbers to be converted to 24,6,0 form must be written as a signed decimal fraction which is less than 1.0 and not less than 0.1 followed by a single signed decimal digit indicating the actual position of the decimal point. That is, any number N is written in the form $N = X.10^Y$, with $1>|x|\geq .1$ and $-9\leq Y\leq 9$, and with X having at most 8 decimal digits. For example,

the number 300, which equals $.3 \times 10^3$, is written as +.3 | +3;

the number $.01\pi$, which equals $.31415927 \times 10^{-1}$ is written as +.31415927 | -1

the number -1/128, which equals $-.78125 \times 10^{-2}$ is written as -.78125 | -2

Alternatively, any number may be converted to 24,6,0 binary form by hand and written as 2 standard single length octal numbers. The procedure for converting by hand is described elsewhere.

Allocation of storage locations to the necessary 2-register numbers, (both for the main program and the subroutines), temporary storage, the main program, the subroutines, and the interpretive subroutine PA 2.2 must at present follow a rather inflexible rule because of the input conversion procedures currently in use. The scheme to be followed is shown diagrammatically below, with decimal addresses used throughout. Notice that parameter x is at present assigned the value 852 in all programs.

Numbers designated by programmer	Storage registers	Comments
address at start of program, usually 32.	main program 2-register numbers, 1st halves	the assignments to consecutive locations of the 2-register constants needed by individual subroutines is handled automatically by the conversion program. The number of locations needed is the sum of the numbers needed by individual subroutines.
total number of locations = k = parameter x2.	subroutine 2-register numbers, 1st halves	
address of start of temporary storage = parameter 0.	temporary storage, 1st halves	the number of temporary locations needed is the maximum of the numbers needed by the main program and the subroutines. Note that all locations are 2-register locations. For 1-register temporary storage, both halves of any 2-register location n may be used by referring to nt for the first half and to ntax2 for the second half.
	main program 2-register numbers, 2nd halves	address of 2nd half of last main program number must be less than 530.
	subroutine 2-register numbers, 2nd halves	
	temporary storage, 2nd halves	
address of start of main program and of each subroutine and address of first instruction to be performed must be indicated	main program	
	subroutines	address of last word of last subroutine must be less than 704.
		space available for print subroutine OT 102.1
address of start of interpretive subroutine = 852 = parameter x	interpretive subroutine	

The interpreted instruction code of this subroutine is given below. The instructions have the same binary value as the similar Whirlwind instructions. Hence they are written, typed and converted in the same way as Whirlwind instructions and are in fact indistinguishable from them. The term "number in location n" is used to signify the number represented in 24,6,0 form by the 32 binary digits contained in the pair of registers n and n+k. The term "register m" is used to signify the single register m. Figures in parentheses give the number of Whirlwind instructions required to interpret the indicated instructions.

Interpreted Instructions	Function
ca n (38)	Clear the MRA and add into it the number in location n.
cs n (36)	Clear the MRA and subtract from it the number in location n.
cm n (37)	Clear the MRA and add into it the magnitude of the number in location n.
ad n (72)	Add the number in the MRA to the number in location n and leave the sum in the MRA.
su n (76)	Subtract from the number in the MRA the number in location n and leave the difference in the MRA.
mr n (49)	Multiply the number in the MRA by the number in location n and leave the product in the MRA.
dv n (74)	Divide the number in the MRA by the number in location n and leave the quotient in the MRA.
ts n (48)	Transfer the number in the MRA to location n.
ex n (48)	Exchange the number in the MRA with the numbers in location n.
sp m (25)	Interpret next the instruction in register m (unless m = ax, in which case transfer control to the register following the one which contains the spax so that the instruction following the spax is performed using the Whirlwind code).
cp m (24)	If the contents of the MRA is a negative number, proceed as in sp n above; if positive, ignore this instruction.
ta m (22)	Transfer the address p + 1 into the right 11 digit positions of register m, leaving the left 5 digit positions unchanged; p being the address of the most recently interpreted sp or effective cp operation.

TITLE: Operations on Real (24,6,0) Floating Point Double Register Numbers (General Subroutine) LSR# PA 2.2 3

Instruction Code and Operation Times:

ts	48	cp	21(+),27(-)	cs	36	cm	37
ta	22	sp	25	ad	72	mr	49
ex	48	ca	38	su	76	dv	74

Preset Parameters (to be typed in program title)

vx2/pk: k=separation in storage of two registers of number
vx/pN: N=address in storage of initial register of this subroutine

Enter→00	ta 179r	Set address of 1^{st} instruction to be interpreted	
01	sp 179r		
02	(p0)	x_1	Multiple register accumulator
03	(p0)	x_1'	
04	(p0)	y_1	
196r→05	sr*30	"ca"	
06	ca ax		
13r,196r→07	ca 191r	"cs"	
08	sp 95r		
196r→09	sp 129r	"ad"	
10	p29		
196r→11	ts 97r	"su"	
12	sp 126r		
196r→13	sp 7r	"cm"	
(170r)14	(p0)	Temporary digits storage	
24r→15	sa 3r	Add two minor products	
16	ts 3r		
17	ca 0	Store overflow	
18	ex 198r		
19	mh 2r	Form major product	
20	sp 158r		
49r,196r→21	mr 2r	"mr" Form two minor products	
22	ex 3r		
23	mr 198r		
24	sp 15r		

196r→25	ex 198r	"dv"
26	ts 97r	
27	cs 102r	Form exponent of $2^{-2}/x_2$.
28	ad 54r	
29	ts 102r	
30	cs 97r	Form and store $-\frac{2^{-2}x_2'}{x_2^2}$
31	mh 97r	
32	ex 198r	
33	sr *2	
34	dv 198r	
35	sl *15	
36	ts 151r	
37	ca 72r	Form and store $\frac{2^{-2}}{x_2^2}$
38	dv 97r	
39	sl *15	
40	ts 198r	
41	mh 97r	Form $\left(\frac{2^{-2}}{x^2}\right)'$ (Use Euclid's algorithm)
42	su 72r	
43	sl *15	
44	su 17r	
45	ad 50r	
46	dv 97r	
47	sl *15	
48	ad 151r	Add two minor parts of reciprocal.
49	sp 21r	

```
50          p1
196r→51     sp 73r    "ts"
111r→52     cs 2r     } Complement x1
53          sp 164r   }
54          p2
196r→55     ca 201r "ta" } Store digits
(181r)56    td(0)        } in indicated
57          sp 178r      } address
119r→58     ao 2r     } Increase x1
59          sp 167r   } by 2^-15
60          ||sp 35r
196r→61     sp 73r    "ex"
62          p63
196r→63     cs 2r     "cp" } Is x1 negative?
64          cp 178r        }
196r→65     ao 179r   "sp" } Set return address
66          td 201r        } for sp ax.
67          ca 180r        } Set pick up order
68          td 179r        } for ordinary cp & sp
69          su 6r
70          cp 199r   } Test to see whether
71          sp 179r   } instruction is sp ax
72          0.20000
51r,61r→73  ca 3r     } Round off x1' and
74          sr 6      } store x1' x 2^-6
75          ts 3r     }
76          sr *9     } Add round-off carry
77          sa 2r     } into x1
78          ts 2r     }
79          ca 0      } Is there an overflow?
80          cp 83r    }
81          su 0      }
82          cp 86r    }
```

```
80r→83      sl 14     } Add overflow
84          ts 2r     } into x1 and x1'
85          ao 4r     } Increase y1.
82r→86      cm 4r     }
87          su 62r    } |y1| -63 > 0?
88          cp 93r    }
89          cs 4r     } y1 ≤ 0?
90          cp 202r   } (i.e. y1 < -63?)
91          cs 62r    } Set y1 = -63
92          ts 4r     }
88r→93      ca 97r    } ts n + k
94          ad 197r   } or ex n + k
8r→95       ts 102r   Store ts,ex,ca,cs,or cm n+k
96          ca 2r     }
(180r)97    (p0)      }
98          ex 3r     } Perform
99          sr *9     } ts,ex,ca,cs, or cm
100         ex 4r     }
101         sl *9     }
(95r)102    (p0)      }
103         sr *9     }
104         ex 3r     }
105         ts 2r     }
106         sl *15    }
107         ex 3r     }
108         sp 177r   }
166r→109    cm 2r     }
110         su0       } x1 ≠ 0?
111         cp 52r    }
112         cm 3r     }
113         su 0      } x1' ≠ 0?
114         cp 122r   }
115         su 50r    } Form
```

Entry	Loc.	Instruction	Remarks
	116	sd 17r	$\lvert x_1' \rvert - 1$
	117	ts 3r	
	118	ca 2r	$x_1 > 0$?
	119	cp_58r	
	120	su 50r	Form
	121	ts 2r	$x_1 - 2^{-15}$
114r→	122	cs 3r	Complement x_1'
	123	ts 3r	
	124	mr 2r	Form $x_1 \cdot x_1'$
	125	sp 166r	
12r→	126	cs 198r	Complement
	127	ts 198r	x_2, x_2'.
	128	cs 97r	
9r→	129	ex 102r	Form and store
	130	su 4r	$y_2 - y_1$
	131	ts 97r	
	132	cp_141r	$y_2 - y_1 > 0$?
	133	ad 4r	Interchange
	134	ts 4r	(x_1, x_1', y_1)
	135	ca 2r	and (x_2, x_2', y_2)
	136	ex 198r	
	137	ts 2r	
	138	ca 3r	
	139	ex 102r	
	140	ts 3r	
132r→	141	cm 97r	$\lvert y_2 - y_1 \rvert - 29 > 0$?
	142	su 10r	
	143	cp_145r	
	144	sp 178r	No need for addition
143r→	145	ad 5r	Store
	146	ts 151r	sr *1 + $\lvert y_2 - y_1 \rvert$

Entry	Loc.	Instruction	Remarks
	147	ca 50r	Set $y_2 = 1$
	148	ex 102r	
	149	sr *15	Form and store
	150	ad 198r	$(x_2 + x_2' \cdot 2^{-15}) \cdot 2^{-1} - \lvert y_2 - y_1 \rvert$
(146r)	151	(p0)	
	152	ts 198r	
	153	sl *15	
	154	ex 3r	
	155	sr *15	Form
	156	ad 2r	$(x_1 + x_1' \cdot 2^{-15}) \cdot 2^{-1}$
	157	sr *1	
20r→	158	ts 2r	Store x_1
	159	sl *15	Add x_1, x_1'
	160	sa 3r	and x_2, x_2'
	161	ts 3r	
	162	ca 198r	
	163	ad 2r	
53r→	164	ts 2r	
	165	mr 3r	Does sign x_1 = sign x_1'?
125r→	166	cp_109r	
59r→	167	ca 3r	Scale factor and
	168	sr *15	store x_1, x_1'
	169	ad 2r	
	170	sf 14r	
	171	ts 2r	
	172	sl *15	
	173	ts 3r	
	174	cs 14r	Form exponent
	175	ad 102r	of x_1, x_1'
	176	ad 4r	
108r→	177	ts 4r	

TITLE: Operations on Real (24,6,0) Floating Point Double Register Numbers (General Subroutine) ISR# P4 2.2 t

34r, 144r→	178	ao 179r	Increase address	(181r)	191	ca(0)	y_2 in reg. 102
171r, 200r→	179	ca(0)	Pick up next instruction		192	sr *9	Hold x_2' in AC.
	180	ts 97r	Store instruction		193	ex 102r	
	181	td 56r	and digits		194	ts 198r	
	182	td 189r			195	sl *15	
	183	ad 197r		(188r)	196	(p0)	Go to part of I.S. for particular instruction
	184	td 191r			197	px2	Separation parameter
	185	sr *27	Form sp to address		198	(p0)	Temporary storage
	186	sl *17	for particular	70r→	199	ad 50r	
	187	ad 60r	instruction		200	~~sp 179r~~ sp 600	Does address equal ax?
	188	ts 196r		(66r)	201	(sp 0)	Return to register following sp ax.
(182r)	189	ca(0)	Pick up x_2, x_2' and y_2.	90r→	202	ca 108r	Produce overflow
	190	ts 102r	Store x_2 in reg. 198		203	ad 108r	alarm

[illegible] interpretive routine: start at 500

600 | sp 179 ax
ca 201 ax
su [illegible]
sp [illegible]
sp 201 ax

601 | ca
2 | qp

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Minimal Routine with Sign Agreement, giving 28 Binary Digit Accuracy in mr)			LSR PA 3.1 t Classification Interpretive
No. of Regs. in Subroutine 96	Temp. Regs. used by Sub. ------	Average Time (operations) see description	Max. Time (operations) see description

Preset Parameters (to be inserted in main program)

v8/ pk: k = separation in storage between the two registers of a double register number

vx/ pN: N = address in storage of the first register of the interpretive subroutine

Description

By means of this subroutine various logical and arithmetic operations can be performed upon real (30,0,0) double-register numbers. The numbers are stored in two registers whose addresses in storage differ by a preset parameter k, i.e. if the major half of a double register number is stored in register n, then the minor half is stored in register n+k.

The operations are written in the usual WW instruction code, but the meanings of these operations may differ from the usual ones (see description of instruction code). Any number of these instructions may be performed in sequence by placing an sp ax before the first instruction in the sequence. The instructions in the sequence are then interpreted successively until a change-of-control instruction is reached at which point either another sequence of instructions is interpreted, or, if the change-of-control instruction is an sp ax ordinary WW operation is resumed at the register following the instruction sp ax.

The multiple register accumulator (mra), in which the results of instructions are left, consists of the storage registers 2r and 3r.

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Minimal Routine with Sign Agreement, giving 28 Binary Digit Accuracy in mr)	LSR PA 3.1 t

Instruction	Function	Av. No. Operations
ts n	Transfer the contents of mra to registers n and n+k	22
ta n	If m is the address of the last sp instruction executed by the subroutine, transfer (m+1) to the last 11 digits of register n	18
ex n	Exchange the contents of mra with the contents of n and n+k	22
cp n	If the contents of mra is negative, proceed as in the sp instruction, if the contents are positive disregard the instruction	22
sp n	If sp n = sp ax, take the next instruction to be interpreted from register n, if sp n = sp ax, resume ordinary WW operation at the register following the instruction sp ax	22
ca n	Clear mra, put the contents of registers n and n+k in mra	23
cs n	Clear mra, put the complement of the contents of registers n and n+k in mra	22
ad n	Add the contents of mra to the contents of registers n and n+k and store the result in mra.	40
su n	Subtract the contents of registers n and n+k from the contents of mra, store the result in mra.	44
cm n	Clear mra, put the absolute value of the contents of registers n and n+k in mra	22
mr n	Multiply the contents of mra by the contents of registers n and n+k, store the result in mra.	34

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Minimal Routine with Sign Agreement, giving 28 Binary Digit Accuracy in mr)	LSR PA 3.1 t

Notes:

1. Entering and Leaving Subroutine

Both entering and leaving the subroutine are accomplished by the instruction sp ax, where vx, a preset parameter, is the address in storage of the first register of the interpretive subroutine. When used to enter the subroutine, the first instruction interpreted is that following the instruction sp ax in storage. When used to leave the subroutine, ordinary WW operation is resumed at the register following the instruction sp ax.

2. Accuracy

All of the operations executed by the subroutine are carried out with a 30 binary digit accuracy except mr, which is carried out with 28 binary digit accuracy.

3. Sign Agreement

A sign agreement routine has been incorporated into the interpretive subroutine. This means that the major and minor halves of the number contained in the mra always have the same algebraic sign after an instruction has been executed by the interpretive subroutine.

4. Reasons for Machine Stoppage During the Subroutine

(a) Arithmetic overflow at 55r - sp n+k 1, i.e. an excessive address is being used for storing the minor half of a double register constant.

(b) Arithmetic overflow at 71r or 72r - overflow in sum or difference for the interpreted instructions ad or su.

(c) Various alarms can result from trying to interpret an instruction which was not meant to be interpreted by the subroutine.

5. In any of the operations the address n may be the address of the mra.

6. The mra consists of the registers 2r and 3r.

7. If k = 1, the two parts of the double register number are stored in consecutive storage registers.

8. No subroutine using preset parameter v8 should be used unless it is later reset.

TITLE: Operations on Real (30, 0, 0) Fixed-Point Double Register Numbers (Minimal Routine with Sign Agreement, giving 28 Binary Digit Accuracy in mr) LSR# PA 3.1 t

Abstract: This subroutine is a (30, 0, 0) interpretive subroutine which performs the instructions ts, ta, ex, cp, sp, ca, cs, ad, su, cm and mr. The double register constants dealt with by the subroutine are in registers whose addresses in storage differ by a preset parameter k, i.e. the C(n, n+k) represents a double register number. Exit and entry to the subroutine are accomplished by the instruction sp ax. In leaving the subroutine, if C(m) = sp ax, then ordinary WW operation is resumed at register m+1. In the description given below, the subroutine is assumed to be executing the instruction C(m) = xx n. There are two preset parameters.

Preset Parameters: (to be inserted in main program)

v8 pk: k = the amount by which the halves of a double register number are separated in storage

vx pN: N = the address in storage of the first register of the interpretive subroutine

	00	ta 51r	Enter interpretive subroutine
	01	sp 51r	
	02	(p0)	mra
	03	(p0)	
64r--→	04	sr 19r	ca
64r--→	05	sp 44r	cs
64r--→	06	sp 69r	ad
64r--→	07	sp 65r	su
64r--→	08	sp 44r	cm
	09	p1	
	10	p0a8	v1
	11	td ax	vx
64r--→	12	mr 2r	
	13	ex 3r	
	14	mr 28r	
	15	sa 3r	
	16	ts 3r	
	17	ca 0	
	18	ex 2r	mr
	19	mh 28r	
	20	ts 28r	
	21	sl 15	
	22	sp 69r	
87r---→	23	ao 2r	Form C(2r) + 2^{-15}
	24	sp 50r	
79r---→	25	cs 2r	Complement C(2r)
	26	sp 49r	
64r---→	27	sp 44r	ts
(62r)	28	(p0)	
64r---→	29	ca 43r	ta
(53r)	30	td (0)	Transfer m+1 to digit
	31	sp 50r	section of register n
64r---→	32	sp 44r	ex
64r---→	33	sp 93r	cp
95r, 64r---→	34	ao 51r	sp
	35	td 43r	Store sp(m+1) in 43r

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Minimal Routine with Sign Agreement, giving 28 Binary Digit Accuracy in mr)   LSR# PA 3.1 t

```
                 36  ca 30r    } Store ca n in 51r
                 37  td 51r    }
                 38  su 11r    }
                 39  cp 41r    } Is xx n = sp ax?
                 40  sp 51r    }
        39r --> 41  ad 9r      }
                 42  cp 51r    }
          (35r) 43  sp (0)       Leave interpretive sub.
5r,8r,27r,32r --> 44  ca 3r    }
          (56r) 45  (p0)       } Perform
                 46  ts 3r     } ts,ex,ca,cs,cm
                 47  ca 2r     }
          (54r) 48  (p0)       }
        26r --> 49  ts 2r      }
24r,31r,76r,94r --> 50  ao 51r
(0r),1r,40r,42r --> 51  ca (0)
                 52  td 61r    } Store n
                 53  td 30r    }
                 54  ts 48r      Store xx n
                 55  ad 10r
                 56  ts 45r      Store xx (n+k)
                 57  td 63r      Store (n+k)
                 58  sr *11    } Set up entry
                 59  ad 4r     } into table
                 60  td 64r    }
          (52r) 61  ca (0)     } Store C(n) in 28r
                 62  ts 28r    }
          (57r) 63  ca (0)       Put C(n+k) in AC
          (60r) 64  sp (0)       Enter table
         7r --> 65  ts 45r     }
                 66  cs 28r    }
                 67  ts 28r    } su
                 68  cs 45r    }
     6r,22r --> 69  sa 3r      }
                 70  ts 3r     }
                 71  ca 2r     } ad,su,mr
                 72  ad 28r    }
                 73  ts 2r     }
        92r --> 74  cs 2r      } Do C(2r) and C(3r)
                 75  mr 3r     } disagree in sign?
                 76  cp 50r    }
                 77  cm 2r     } Is C(2r) ≠ 0?
                 78  su 0      }
                 79  cp 25r    }
                 80  cm 3r     } Is C(3r) ≠ 0?
                 81  su 0      }
                 82  cp 90r    }
                 83  su 9r     } Form 1 -|C(3r)|
                 84  ad 17r    }
                 85  ts 3r     }
                 86  ca 2r     } Is C(2r) positive?
                 87  cp 23r    }
                 88  su 9r     } Form C(2r) - 2^-15
                 89  ts 2r     }
        82r --> 90  cs 3r      } Complement C(3r)
                 91  ts 3r     }
                 92  sp 74r      Re-enter sign agreement
        33r --> 93  cs 2r      } cp
                 94  cp 50r    } Is C(2r) negative?
                 95  sp 34r      Perform sp order.
```

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Operations on Real (30,0,0) Fixed-Point Double-Register Numbers (General Routine with Sign Agreement, No Division) (Interpretive)			LSR PA 3.5t TAPE T721-2 Classification Interpretive
No. of Regs. in Subroutine 123	Temp. Regs. used by Subroutine ----------	Average Time (operations) see description	Max. Time (operations) see description

Preset Parameters

vx2/ pk: k = separation in storage between the two registers of a double register number

vx/ pN: N = address in storage of the first register of the interpretive subroutine

Description

By means of this subroutine various logical and arithmetic operations can be performed upon real (30,0,0) double-register numbers. The numbers are stored in two registers whose addresses in storage differ by a preset parameter k, i.e. if the major half of a double register number is stored in register n, then the minor half is stored in register n+k.

The operations are written in the usual WW instruction code, but the meanings of these operations may differ from the usual ones (see description of instruction code). Any number of these instructions may be performed in sequence by placing an sp ax before the first instruction in the sequence. The instructions in the sequence are then interpreted successively until a change-of-control instruction is reached at which point either another sequence of instructions is interpreted, or, if the change-of-control instruction is an sp ax ordinary WW operation is resumed at the register following the instruction sp ax.

The multiple register accumulator (mra), in which the results of instructions are left, consists of the storage registers 2r and 3r.

Instruction	Function	Av. No. Operations
ts n	Transfer the contents of mra to registers n and n+k	22
ta n	If m is the address of the last sp instruction executed by the subroutine, transfer (m+1) to the last 11 digits of register n	18
ex n	Exchange the contents of mra with the contents of n and n+k	22
cp n	If the contents of mra is negative, proceed as in the sp instruction, if the contents are positive disregard the instruction	22

TITLE: Operations on Real (30,0,0) Fixed-Point Double-Register Numbers (General Routine with Sign Agreement, No Division) (Interpretive) LSR PA 3.5t

Instruction	Function	Av. No. Operations
sp n	If sp n = sp ax, take the next instruction to be interpreted from register n, if sp n = sp ax, resume ordinary WW operation at the register following the instruction sp ax	22
ca n	Clear mra, put the contents of registers n and n+k in mra	23
cs n	Clear mra, put the complement of the contents of registers n and n+k in mra	22
ad n	Add the contents of mra to the contents of registers n and n+k and store the result in mra	40
su n	Subtract the contents of registers n and n+k from the contents of mra, store the result in mra	44
cm n	Clear mra, put the absolute value of the contents of registers n and n+k in mra	22
mr n	Multiply the contents of mra by the contents of registers n and n+k, store the result in mra.	34
sl(800+n)	Multiply C(mra) by 2^n and leave the result in the mra	29
sr(800+n)	Multiply C(mra) by 2^{-n}, and store the first 30 digits of the result without roundoff in the mra.	28

Notes

1. Entering and Leaving Subroutine

Both entering and leaving the subroutine are accomplished by the instruction sp ax, where vx, a preset parameter, is the address in storage of the first register of the interpretive subroutine. When used to enter the subroutine, the first instruction interpreted is that following the instruction sp ax in storage. When used to leave the subroutine, ordinary WW operation is resumed at the register following the instruction sp ax.

2. Accuracy

All of the operations executed by the subroutine are carried out with 30 binary digit accuracy, affected only by roundoff on the thirty-first digit.

3. Sign Agreement

A sign agreement routine has been incorporated into the interpretive subroutine. This means that the major and minor halves of the number contained in the mra always have the same algebraic sign after an instruction has been executed by the interpretive subroutine.

4. Reasons for machine stoppage during the subroutine

(a) Arithmetic overflow at 56r - sp n+k, an excessive address is being used for storing the minor half of a double register constant.

TITLE: Operations on Real (30,0,0) Fixed-Point Double-Register Numbers (General Routine with Sign Agreement, No Division) (Interpretive) LSR PA 3.5t

(b) Arithmetic overflow at 72r or 73r - overflow in sum or difference for the interpreted instructions ad or su.

(c) Various alarms can result from trying to interpret an instruction which was not meant to be interpreted by the subroutine.

5. The mra consists of registers 2ax (at 2r) and 3ax (at 3r).

FCH	AS
12/5/51	2/5/52

TITLE: Operations on Real (30,0,0) Fixed-Point Double-Register Numbers (General Routine with Sign Agreement, No Division) (Interpretive)

LSR# PA 5.5t
TAPE# T721-2

Abstract: This subroutine is a (30,0,0) interpretive subroutine which performs the instructions ts, ta, ex, cp, ca, cs, ad, su, cm, mr, sl, and sr. The double register constants dealt with by the subroutine are in registers whose addresses in storage differ by a preset parameter k, i.e. the C(n,n+k) represents a double register number. Exit and entry to the subroutine are accomplished by the instruction sp ax. In leaving the subroutine, if C(m) C(m) = sp ax, then ordinary WW operation is resumed at register m+1. In the description given below, the subroutine is assumed to be executing the instruction C(m) = xx n. There are two preset parameters.

Preset Parameters: (to be inserted in main program)

vx2: pk: k = the amount by which the halves of a double register number are separated in storage

vx: pN: N = the address in storage of the first register of the interpretive subroutine

	00	ta 51r	Set next instruction address
	01	sp 51r	
(23r)→	02	(p0)	x_2 mra
	03	(p0)	x_2^1
65r →	04	sr 19r	ca
65r →	05	sp 44r	cs
65r →	06	sp 70r	ad
65r →	07	sp 66r	su
65r →	08	sp 44r	cm
33r →	09	cs 2r	Is C(2r) negative?
	10	cp 50r	
	11	sp 34r	Perform sp instruction
65r →	12	ts 19r	mr
	13	mr 3r	
	14	sp 94r	
65r →	15	ca ax	sl (vx)
65r →	16	ca 3r	sr
	17	sr* 15	sl, sr $C(n+k)=x_2^1$
	18	ad 2r	
(103r)(55r)(12r)→	19	(p0)	

	20	ts 2r	
	21	sl* 15	
	22	sp 92r	
88r→	23	ao 2r	Form $C(2r)+2^{-15}$
	24	sp 50r	
80r→	25	cs 2r	Complement C(2r)
	26	sp 49r	
65r→	27	sp 44r	ts
(68r)(63r)	28	(p0)	$C(n) = x_2$
65r→	29	ca 43r	ta
(53r)	30	td(0)	Transfer m+1 to digit section of register D
	31	sp 50r	
65r→	32	sp 44r	ex
65r→	33	sp 9r	cp
11r,65r→	34	ao 51r	sp
	35	td 43r	store sp(m+1) in 43r
	36	ca 62r	Is xxn = spax
	37	td 51r	
	38	su 15r	
	39	cp 41r	

TITLE: Operations on Real (30,0,0) Fixed-Point Double-Register Numbers (General Routine with Sign Agreement, No Division) (Interpretive)

LSR# PA 3.5t
TAPE# T721-2

Entered from	Loc.	Instruction	Remarks
	40	sp 51r	Leave interpretive subroutine (40–43)
39r →	41	ad 122r	
	42	op 51r	
(35r)	43	sp 0	
5r, 8r, 27r, 32r →	44	ca 3r	Perform ts, ex, ca, cs, and cm. (44–49)
(97r)(66r)(57r)(113r)	45	(p0)	
	46	ts 3r	
	47	ca 2r	
(54r)	48	(p0)	
26r →	49	ts 2r	
10r, 24r, 31r, 77r →	50	ao 51r	Pick up next instruction to be interpreted (50–51)
1r, 40r, 42r, (0r), (34r)(37r)(50r) →	51	ca(0)	
	52	td 62r	Store n (52–53)
	53	td 30r	
	54	ts 48r	Store xxn (54–55)
	55	ts 19r	
	56	ad 121r	
	57	ts 45r	Store xx(n+k)
	58	td 64r	Store (n+k)
	59	sr* 11	Set up entry into table (59–61)
	60	ad 4r	
	61	td 65r	
(52r)	62	ca(0)	Store C(n) in 28r (62–63)
	63	ts 28r	
(58r)	64	ca(0)	put C(n+k) = x'_2 in
(61r)	65	sp(0)	enter table
7r →	66	ts 45r	su (66–69)
	67	cs 28r	
	68	ts 28r	
	69	cs 45r	
6r, 120r →	70	sa 3r	ad, su, mr (70–74)
	71	ts 3r	
	72	ca 2r	
	73	ad 28r	
	74	ts 2r	
93r →	75	cs 2r	Do C(2r) and C(3r) agree in sign (75–77)
	76	mr 3r	
	77	op 50r	
	78	cm 2r	Is C(2r) ≠ 0? (78–80)
	79	su0	
	80	cp 25r	
	81	cm 3r	Is C(3r) ≠ 0? (81–83)
	82	su0	
	83	cp 91r	
	84	su 122r	Form $-1 + \|C(3r)\|$ (84–86)
	85	ad 118r	
	86	ts 3r	
	87	ca 2r	Is C(2r) positive? (87–88)
	88	cp 23r	
	89	su 122r	Form $C(2r) - 2^{-15}$ (89–90)
	90	ts 2r	
83r →	91	cs 3r	Complement C(3r) (91–92)
22r →	92	ts 3r	
	93	sp 75r	Re-enter sign agreement
AC 14r →	94	sr 2	Form $\frac{1}{4} x'_1 x'_2$ and store in 19r (94–95)
	95	ex 19r	
	96	mh 2r	Form $x_1 x'_2$, store major half in 45r (96–97)
	97	ts 45r	
	98	sl*15	Form $\frac{1}{4} [x_1 x'_2]'$ (98–99)
	99	sr 2	

TITLE: Operations on Real (30,0,0) Fixed-Point Double-Register Numbers (General Routine with Sign Agreement, No Division)(Interpretive)

LSR# PA 3.5t
TAPE # T721-2

100	ad 19r	Accumulate round-off and store in 2r
101	ex 2r	
102	mh 28r	Form x_1x_2, store major half in 19r and minor half in 28r
103	ts 19r	
104	sl* 15	
105	ex 28r	
106	mh 3r	Form $x_1' x_2$ and store major half in 3r
107	ts 3r	
108	sl* 15	Form $\frac{1}{4}\left[x_1' x_2\right]'$
109	sr 2	
110	ad 2r	Form total round-off and add into major half of x_1x_2'
111	sr 13	
112	sa 45r	
113	ts 45r	
114	ca 19r	
115	ex 28r	Add x_1x_2' and the major half of x_1x_2'
116	sa 45r	
117	ts 2r	
118	ca0	
119	ex 2r	
120	sp 70r	Enter addition routine
121	p0ax2	v1
122	p1	

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Short, fast routine without sign agreement giving 28 binary digit accuracy in mr) Interpretive	LSR PA 3.10t TAPE T798 Classification Interpretive

No. of Regs. in Subroutine	Temp. Regs. used by Sub.	Average Time (operations)	Max. Time (operations)
78	0	----	----

Preset Parameters (To be inserted in title of main program)

vx pN, where N is the address in storage of the first register of the interpretive subroutine

vx2 pk, where k is the separation in storage between the two registers of a double register number

Description

By means of this subroutine various logical and arithmetic operations can be performed upon real (30,0,0) double-register numbers. The numbers are stored in two registers whose addresses in storage differ by a preset parameter k, i.e. if the major half of a double register number is stored in register n, then the minor half is stored in register n+k.

The operations are written in the usual WW instruction code, but the meanings of these operations may differ from the usual ones (see description of instruction code). Any number of these instructions may be performed in sequence by placing an sp ax before the first instruction in the sequence. The instructions in the sequence are then interpreted successively until a change-of-control instruction is reached at which point either another sequence of instructions is interpreted, or, if the change-of-control instruction is an sp ax ordinary WW operation is resumed at the register following the instruction sp ax.

The multiple register accumulator (MRA), in which the results of instructions are left, consists of the storage registers 2r and 3r.

This interpretive subroutine does not contain a sign agreement routine and hence the contents of the MRA may have different signs, e.g. in the subtraction

$$10\cdot 2^{-15} + 5\cdot 2^{-30} - \left[5\cdot 2^{-15} + 10\cdot 2^{-30}\right] = 5\cdot 2^{-15} + (-10)\cdot 2^{-30}$$

the result has major and minor halves of unlike sign. This fact must be remembered by the programmer if computations are carried on outside of the subroutine or if other library subroutines are interpreted by the interpretive subroutine. In the latter case, a library subroutine should not be used unless it is explicitly stated in its specification sheet that it is permissable to use it with interpretive subroutines not containing sign agreement.

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Short, fast routine without sign agreement giving 28 binary digit accuracy in mr) Interpretive

LSR PA 3.10t
TAPE T798

Instruction	Function	Av. No. Operations
ts	Transfer the contents of **MRA** to registers n and n+k	17
ta	If m is the address of the last sp instruction executed by the subroutine, transfer (m+1) to the last 11 digits of register n	13
ex	Exchange the contents of **MRA** with the contents of n and n+k	17
cp	If the contents of **MRA** is negative, proceed as in the sp instruction, if the contents are positive disregard the instruction	(+)14 (-)25
sp	If sp n ≠ sp ax, take the next instruction to be interpreted from register n, if sp n = sp ax, resume ordinary WW operation at the register following the instruction sp ax	18
ca	Clear MRA, put the contents of registers n and n+k in **MRA**	17
cs	Clear **MRA**, put the complement of the contents of registers n and n+k in **MRA**	17
ad	Add the contents of **MRA** to the contents of registers n and n+k	18
su	Subtract the contents of registers n and n+k from the contents of MRA	17
mr	Multiply the contents of **MRA** by the contents of registers n and n+k, store the result in **MRA**	33

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Short, fast routine without sign agreement giving 28 binary digit accuracy in mr) Interpretive

LSR PA 3.10t

TAPE T798

Notes:

1. Entering and Leaving Subroutine

Both entering and leaving the subroutine are accomplished by the instruction sp ax, where vx, a preset parameter, is the address in storage of the first register of the interpretive subroutine. When used to enter the subroutine, the first instruction interpreted is that following the instruction sp ax in storage. When used to leave the subroutine, ordinary WW operation is resumed at the register following the instruction sp ax.

2. Accuracy

All of the operations executed by the subroutine are carried out with a 30 binary digit accuracy except mr, which is carried out with 28 binary digit accuracy.

3. Sign Agreement

PA 3.10t does not contain sign agreement. Hence the major and minor halves of a number need not have the same sign.

4. Reasons for Machine Stoppage During the Subroutine

(a) Arithmetic overflow at 56r--an excessive address is being used for storing the minor half of a double register number.

(b) Arithmetic overflow at 49r or 50r--overflow in sum or difference for the interpreted instructions ad or su.

(c) Various alarms can result from trying to interpret an instruction which was not meant to be interpreted by the subroutine.

5. The MRA consists of registers 2ax(2r) and 3ax(3r).

FCH	MAS
2/5/52	2/8/52

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Short, fast routine without sign agreement giving 28 binary digit accuracy in mr) Interpretive

LSR PA 3.10t

TAPE T-798

Abstract: This subroutine is a (30,0,0) interpretive subroutine which performs the instructions ts, ta, ex, cp, sp, ca, cs, ad, su, cm and mr. The double register constants dealt with by the subroutine are in registers whose addresses in storage differ by a preset parameter k, i.e. the C(n, n+k) represents a double register number. Exit and entry to the subroutine are accomplished by the instruction sp ax. In leaving the subroutine, if C(m) = sp ax, then ordinary WW operation is resumed at register m+1. In the description given below, the subroutine is assumed to be executing the instruction C(m) = xx n. There are two preset parameters. The subroutine does not contain sign agreement. Hence the major and minor halves of a double register number can have unlike signs.

Preset Parameters

vx pN, where N is the address in storage of the first register of the interpretive subroutine

vx2 pK, where K is the separation in storage between the two registers of a double register number

	00	ta 53r	Enter interpretive subroutine
	01	sp 53r	
(76r)	02	(p0)	mra
(74r)	03	(p0)	
59r →	04	sp 44r	ca
59r →	05	sp 44r	cs
59r →	06	sp 7r	ad
6r, 59r →	07	ad 26r	su; Form ca(n+k) or cs (n+k) & store in 9r
	08	ts 9r	
77r → (8r)(15r)	09	(p0)	Form $x_2^1 + x_1^1$ and store in 3r (ad, su)
	10	sa 3r	
	11	sp 48r	Form $x_1 x_2^1 + x_1 x_2^1 + x_o^1 x_2$ (mr)
39r →	12	td 68r	mr; Set address at 68r to n & store mrn in 72r
	13	ts 72r	
	14	ad 25r	Store mr(n+k) in 9r
	15	ts 9r	
	16	ca 51r	Store ts2r in 50r
	17	ts 50r	
	18	sp 67r	
33r →	19	cm 2r	Is C(2r) ≠ 0
	20	su 0	
	21	cp 60r	
	22	cs 2r	Is C(2r) or C(3r) pos.?
64r →	23	cp 52r	
	24	sp 34r	Perform interpreted sp
	25	pa x2	
	26	1.67777ax2	
50r →	27	sp 44r	ts
	28	sp ax	
59r →	29	td 65r	ta; Transfer n to digit section of reg. 65r
	30	ca 42r	
	31	sp 65r	
59r →	32	sp 44r	ex
59r, 62r →	33	sp 19r	cp
24r, 59r →	34	ao 53r	sp
	35	td 42r	Store sp (m+1) in 42r
	36	ca 50r	Store ca n in 53r
	37	td 53r	
	38	su 28r	Is xxn = sp ax?
	39	ts 47r	

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Short, fast routine without sign agreement giving 28 binary digit accuracy in mr) Interpretive

LSR PA 3.10t
TAPE T798

```
                    40  cm 47r      }
                    41  su 0        }
             (35r)  42  cp(0)       }
                    43  sp 53r      }
4r,5r
27r,32r  -->        44  ad 25r      } Store xx(n+k) in 47r
                    45  ts 47r      }
                    46  ca 3r       }
(39r)
(45r)        (69r)  47  (p0)        }
11r      -->        48  ts 3r       } Perform ts,ex,ca,cs,
                    49  ca 2r       }   ad,su,mr
   (17r)(54r)       50  (p0)        }
                    51  ts 2r       }
23r,66r  -->        52  ao 53r      } Pick up instruction
(34r)(37r)(0r)      53  ca(0)       }  to be interpreted
1r,43r   -->
(52r)               54  ts 50r        Store xxn
                    55  sr* 11      }
                    56  ad 33r      } Set up entry into
                    57  ts 59r      }    table
                    58  ca 50r        Pick up xxn
```

```
(57r)      59  (p0)         Enter table
21r  -->   60  cm 3r      }
           61  su 0       } Is C(3r) ≠ 0
           62  cp 34r     }
           63  cs 3r
           64  sp 23r
(29r)
31r  -->   65  td (0)     } Transfer m+1 to digit section
           66  sp 52r     }   of register n
18r  -->   67  ca 2r        Form x_1 x_2
           68  mh(0)n
           69  ts 47r       Store [x_1 x_2] in 47r
           70  sl 15      } Store [x_1 x_2]^1 in 3r
           71  ex 3r      }
(13r)      72  (p0)         Form [x_1^1 x_2]
           73  sa 3r      } Form [x_1 x_2]^1 + [x_1^1 x_2]
           74  ts 3r      }   and store in 3r
           75  ca 47r     } Store [x_1 x_2] in 47r
           76  ex 2r      }
           77  sp 9r
```

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Operations on real (15,0,c) fixed-point single register numbers (Interpretive)			LSR PA 8.1t TAPE T724-2 Classification Interpretive
No. of Regs. in Subroutine 47	Temp. Regs. used by Sub. None	Average Time (operations)	Max. Time (operations)
		See description of order code.	

Preset Parameters (To be typed in the title of the Main Program)

vx: pN: N = address in storage of initial register of subroutine
vx3:pc: c = the number of binary digits to the right of the binary point in the (15,0,c) numbers
vx1:pM: M = address in storage of the initial register of the storage block for (15,0,c) numbers

Description

By means of this subroutine various logical and arithmetic operations can be performed on real numbers expressed in the (15,0,c) system, $0 \leq c \leq 15$. The (15,0,c) constants are stored in the following manner. Let c be fixed and let a be a positive number such that

$$2^{-(15-c)} \leq a \leq 2^{c+1} - 1$$

Then a can be written as the sum of a c digit binary integer and a binary fraction. The binary fraction is then rounded off to 15 - c digits and the result stored with the sign digit zero. If $a < 0$, repeat this procedure for $-a$, complement this number and store the result.

For example, let $c = 3$ and $a = -3\frac{1}{3}$

Then

$$-a = 3 + \frac{1}{3}$$

and

$$3 = +.011$$

$$\frac{1}{3} = +.\underbrace{010101010101}_{15-3=12 \text{ digits}}010$$

$$3 + \frac{1}{3} = 011.010101010101$$

We now complement the following number

0.011010101010101

and store the following result

1.100101010101010

The programmer need not carry out this conversion process himself but instead need only write the fixed point decimal number. By giving this information to the proper subroutine in the IP section of the library the conversion can be done automatically.

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Operations on real (15,0,c) fixed-point single register numbers (Interpretive)	LSR PA 8.1t TAPE T724-2

Description (continued)

Operations upon numbers are written in the usual Whirlwind instruction code, but the meaning of these operations may differ from the usual ones. Any number of these operations may be performed in sequence by placing an spax before the first instruction in the sequence. The instructions in the sequence are then interpreted successively until a change-of-control instruction is reached (see description of order code) at which point either another sequence of instructions is interpreted, or, if the change of control instruction is an spax ordinary Whirlwind operation is resumed at the register following the instruction spax.

The multiple register accumulator, in which the results of interpreted instructions are left, consists of the storage register 2ax.

Inst.	Function	Average # Operations
ts n	Transfer C(mra) to register n.	14
td n	Transfer the last 11 digits from the mra to the last 11 digits of register n	14
ta n	If m is the address of the last sp instruction or effective cp instruction executed by the subroutine, transfer (m+1) into the last 11 digits of register n	13
ex n	Exchange C(mra) with C(n)	14
cp n	If C(mra) is negative, proceed as in the sp instruction, if C(mra) is positive, disregard the instruction	18
sp n	If spn ≠ spax, take the next instruction to be interpreted from register n, if spn = spax, resume ordinary Whirlwind operation at the register following the instruction spax	21
ca n	Clear mra, put C(n) in the mra	14
cs n	Clear mra, put the complement of C(n) in the mra	14
ad n	Add C(mra) to C(n) and store the result in the mra	14
su n	Subtract C(n) from C(mra) and store the result in the mra	14
cm n	Clear the mra, and put the positive magnitude of C(n) in the mra	14
mr n	Multiply C(mra) by C(n), store the result in the mra	14
sl n	Multiply C(mra) by 2^n and store the result in the mra	14
sr n	Multiply C(mra) by 2^{-n} and store the result in the mra	14

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Operations on real (15,0,c) fixed-point single register numbers (Interpretive)	LSR PA 8.1t TAPE T724-2

Notes:

1. Entering and leaving subroutine
   Both entering and leaving the subroutine are accomplished by the instruction sp ax, where vx, a preset parameter, is the address in storage of the first register of the interpretive subroutine. When used to enter the subroutine, the first instruction interpreted is that following the instruction sp ax in storage. When used to leave the subroutine, ordinary WW operation is resumed at the register following the instruction sp ax.

2. Accuracy
   All of the operations executed by the subroutine are carried out with a 15 binary digit accuracy.

3. Calculation with integers
   If c = 15, the (15,0,c) numbers dealt with by the subroutine are binary integers.

4. Reasons for machine stoppage during the subroutine
   a) Arithmetic overflow at 35r - overflow in a sum or difference for the interpreted instructions ad or su.
   b) Various alarms can result from trying to interpret an instruction which was not meant to be interpreted by the subroutine.

5. Overflow during mr
   It should be noted that overflows can occur during multiplication of (15,0,c) numbers. These will not cause an alarm since the overflow is shifted out of the left hand end of the accumulator, i.e., the product is formed modulo 2^c.

6. In any of the operations the address n may be the address of the mra.

7. The mra consists of register 2r (or 2ax).

FCH	AS
12/6/51	2/5/52

TITLE: Operations on real (15,0,c) fixed-point single register numbers (Interpretive)

LSR# PA 8.1t
TAPE# T 724-2

Abstract: PA 8.1 is a (15,0,c) interpretive subroutine which performs the instructions ts, td, ta, ex, cp, sp, ca, cs, ad, su, cm, mr, sl and sr. Exit and entry to the subroutine are accomplished by the instruction spax. In leaving the subroutine, if C(m) = spax, then ordinary WW operation is resumed at register m + 1.

Preset Parameters:

vx1: M address in storage of the initial register of the storage block for (15,0,C) numbers.

vx3: C where C is the number of binary digits to the left of the binary point

vx N where N is the address in storage of the first register of the interpretive routine

In the description given below the subroutine is assumed to be executing the order C(m) = xxn

```
            00  ta 38r    } Enter interpretive
            01  sp 38r    }   subroutine
            02  (p0)        mra
46r    ---> 03  sr 18r      ca
46r    ---> 04  ca ax       cs
46r    ---> 05  sp 34r      ad
46r    ---> 06  sp 34r      su
46r    ---> 07  sp 34r      cm
32r    ---> 08  cs 2r     }
            09  cp 37r    } cp
            10  sp 16r    }
46r    ---> 11  mh 2r     }
            12  sl0ax3    } mr
            13  sp 36r    }
46r    ---> 14  sp 34r      sl
46r    ---> 15  sp 34r      sr
10r,33r --> 16  ao 38r    } store cp(m+1) in
            17  td 24r    }   24r
            18  ca 45r    } store ca n in 38r
            19  td 38r    }
            20  su 4r     }
            21  ts 35r    }
            22  cm 35r    } Is xxn=spax?
            23  su0       }
(17r)       24  cp0       }

46r    ---> 25  sp 38r
            26  sp 34r      ts
46r    ---> 27  sp 34r      td
46r    ---> 28  ca 24r      ta
(41r)       29  td(0)       Transfer m+1 to digit section
                              of register n
            30  sp 37r
46r    ---> 31  sp 34r      ex
46r    ---> 32  sp 8r       cp
46r    ---> 33  sp 16r      sp
5r,6r,7r        
14r,15r     ---> 34  ca 2r   } Perform ca, cs, ad, su, cm,
26r,27r,31r ---> 35  (p0)    }  sl, sr, ts, td, and ex
(21r)(39r)
13r    ---> 36  ts 2r    }
9r,30r ---> 37  ao 38r
(0r)(16r)
1r, 25r  }  38  ca(0)
(37r)(19r)  39  ts 35r      Store xxn
            40  td 45r   } Store n
            41  td 29r   }
            42  sr* 11   }
            43  ad 3r    } Set up entry to table
            44  td 46r   }
(40r)       45  ca(0)       Put C(n) in AC
(44r)       46  sp(0)       Enter table.
```

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Form $\sin \frac{\pi}{2} x$ from x Stored in AC, and Leave Result in AC			LSR TF 0.1 t Classification Closed
No. of Regs. in Subroutine 19	Temp. Regs. used by Sub. d - 1t	Average Time (operations) 15	Max. Time (operations) 15

Program Parameters
on entering Subroutine
ac: x
ar: return address

Results
on leaving Subroutine

ac: $\sin \frac{\pi}{2} x$

Description

The subroutine gives $\sin \frac{\pi}{2} x = ax-bx^3+cx^5-dx^7$ for $-1 < x < 1$. The subroutine is entered with x in the accumulator and on returning to the main program, $\sin \pi/2\ x$ is in the AC. The maximum error is approximately ± 0.00005 and the average error is ± 0.00003.

TITLE: Form $\sin \frac{\pi}{2} x$ from x Stored in AC, and Leave Result in AC. LSR# TF 0.1 t

Abstract: This subroutine gives $\sin \frac{\pi}{2} x = ax - bx^3 + cx^5 - dx^7$ as a single length fixed point number in the accumulator where $-1 < x < 1$.

Entering the subroutine:
ac: x

Leaving the subroutine:
ac: $\sin \pi/2 \ x$

Temporary Storage:
d unused
1t used to store the value of x

00	ta 14r	Set return address	
01	ts 1t	Store x	
02	mh 1t	x^2	
03	mh 15r	$-dx^2$	
04	ad 16r	$c - dx^2$	
05	mh 1t	$cx - dx^3$	
06	mh 1t	$cx^2 - dx^4$	
07	ad 17r	$-b+cx^2-dx^4$	
08	mh 1t	$-bx + cx^3 - dx^5$	
09	mh 1t	$-bx^2 + cx^4 - dx^6$	
10	sr *1	$(-bx^2+cx^4-dx^6)2^{-1}$	
11	ad 18r	$(a-bx^2+cx^4-dx^6)2^{-1}$	
12	mh 1t	$(ax-bx^3+cx^5-dx^7)2^{-1}$	
13	sl 1	$ax-bx^3+cx^5-dx^7$	
(0r) 14	sp (0)	Return to main program	
15	1.77560	$-d$	
16	0.05055	c	
17	1.26521	$-b$	
18	0.62210	$a \times 2^{-1}$	

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Form cosine $\frac{\pi}{2}$y from y Stored in AC, and Leave Result in AC (15,0,0)	LSR TF 1.1t Tape 705-1 Classification Closed

No. of Regs. in subroutine	Temp. Regs. used by Sub.	Average Time (operations)	Max. Time (operations)
26	d -1t(2)	20	20

Program Parameters
on entering Subroutine
ac: y
ar: Return address

Results
on leaving Subroutine

ac: cosine $\frac{\pi}{2}$y

Description

Gives cosine $\frac{\pi}{2}$y by changing y to x so that $\cos\frac{\pi}{2}y = \sin\frac{\pi}{2}x = ax - bx^3 + cx^5 - dx^7$, for $-1<x<1$. If y=0, there will be an overflow since the cosine of zero is one. The subroutine is entered with y in the accumulator and on returning to the main program cosine $\frac{\pi}{2}$y is in the accumulator. The maximum error is approximately ±0.00005 and the average error ±0.00002.

Notes
1. If y is ±zero, an arithmetic overflow will occur in register 5r.

DMN+DGA	JWC III	MAS
1/14/52	1/16/52	1/22/52

TITLE: Form cosine $\frac{\pi}{2}y$ from y Stored in AC, and Leave Result in AC (15,0,0) — LSR# TF 1.1t, Tape 705-1

Abstract: Gives cosine $\frac{\pi}{2}y$ by forming $x=1-|y|$ so that cosine $\frac{\pi}{2}y$ = sine $\frac{\pi}{2}x$ and evaluating $\sin\frac{\pi}{2}x = ax - bx^3 + cx^5 - dx^7$, $-1<x<1$. If $y=\pm 0$ there will be an overflow at register 5r.

Upon entering the subroutine:
ac: y

Upon leaving the subroutine:
ac: cosine $\frac{\pi}{2}y$

Temporary registers:
d - unused
1t - used to store x

00	ta 19r	Set return address		13	mh 1t	$-bx + cx^3 - dx^5$
01	ts 1t	Transfer y to 1t		14	mh 1t	$-bx^2 + cx^4 - dx^6$
02	cp 4r	y negative? Yes		15	sr*1	$(-bx^2 + cx^4 - dx^6)2^{-1}$
03	cs 1t	No. -y in ac		16	ad 25r	$(a - bx^2 + cx^4 - dx^6)2^{-1}$
2r → 04	ad 20r	$(1 - 2^{-15}) - y$		17	mh 1t	$(ax - bx^3 + cx^5 - dx^7)2^{-1}$
05	ad 21r	Add 2^{-15} } (Possible overflow)		18	sl 1	$ax - bx^3 + cx^5 - dx^7$
06	ts 1t	$1-\lvert y\rvert = x$ in 1t	(0r)	19	sp(0)	Return to main program
07	mh 1t	x^2		20	0.77777	$1 - 2^{-15}$
08	mh 22r	$-dx^2$		21	0.00001	2^{-15}
09	ad 23r	$c - dx^2$		22	1.77560	$-d$
10	mh 1t	$cx - dx^3$		23	0.05055	c
11	mh 1t	$cx^2 - dx^4$		24	1.26521	$-b$
12	ad 24r	$-b + cx^2 - dx^4$		25	0.62210	$a \times 2^{-1}$

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Form Sine $\frac{\pi}{2}x$ from x Stored in AC, and/or Form Cosine $\frac{\pi}{2}y$ from y Stored in AC, Leave Result in AC, (15,0,0)			LSR TF 7.1 t Tape T 750-1 Classification Closed
No. of Regs. in Subroutine 28	Temp. Regs. used by Sub. d - 1t	Average Time (operations) 15 - sine 20 - cosine	Max. Time (operations) 15 - sine 20 - cosine

Program Parameters
on entering Subroutine
ac: x or y
ar: return address

Results
on leaving Subroutine

ac: sine $\frac{\pi}{2}x$ or cosine $\frac{\pi}{2}y$

Description

If this subroutine is entered at register 0r, it will calculate cosine $\frac{\pi}{2}y$ by changing y to x so that cosine $\frac{\pi}{2}y$ = sine $\frac{\pi}{2}x$ and evaluating sine $\frac{\pi}{2}x = ax - bx^3 + cx^5 - dx^7$, $-1<x<1$. If it is entered at register 7r, it will calculate sine $\frac{\pi}{2}x$. There will be an overflow at register 5r if y = 0. The subroutine is entered with either x or y in the accumulator, and on returning to the main program either sine $\frac{\pi}{2}x$ or cosine $\frac{\pi}{2}y$ is in the accumulator. The maximum error is approximately ± 0.00005 and the average error is ± 0.00002.

Notes:
1. Enter at 0r if cosine $\frac{\pi}{2}y$ is desired
   enter at 7r if sine $\frac{\pi}{2}x$ is desired
2. If y = ± 0, an overflow will occur in register 5r.

DMN/DGA	JWC III	DEL
1/14/52	1/16/52	1/18/52

TITLE: FORM SINE $\frac{\pi}{2}x$ FROM x STORED IN AC, AND/OR FORM COSINE $\frac{\pi}{2}y$ FROM y STORED IN AC, LEAVE RESULT IN AC, (15,0,0)

LSR# TF 7.1t
Tape T 750-1

Abstract: If this subroutine is entered at register 0r, it will calculate cosine $\frac{\pi}{2}y$ by changing y to x so that cosine $\frac{\pi}{2}y$ = sine $\frac{\pi}{2}x$ and evaluating sine $\frac{\pi}{2}x = ax-bx^3+cx^5-dx^7$, $-1<x<1$. If it is entered at register 7r, it will calculate sine $\frac{\pi}{2}x$. There will be an overflow at register 5r if y=0.

Upon entering the subroutine:
ac: x or y

Upon leaving the subroutine:
ac: sine $\frac{\pi}{2}x$ or cosine $\frac{\pi}{2}y$.

Temporary registers:
d - unused
1t - used to store x

Cosine ⟶	00	ta 21r	Set return address
	01	ts 1t	y in 1t
	02	cp 4r	Is y negative? Yes
	03	cs 1t	No. -y in ac
2r ⟶	04	ad 22r	$(1-2^{-15})-\lvert y\rvert$ in ac
	05	ad 23r	$1-\lvert y\rvert = x$ in ac Possible overflow
	06	sp 8r	
Sine ⟶	07	ta 21r	Set return address (0r)(7r)
6r ⟶	08	ts 1t	x in 1t
	09	mh 1t	x^2
	10	mh 24r	$-dx^2$
	11	ad 25r	$c-dx^2$
	12	mh 1t	$cx-dx^3$
	13	mh 1t	cx^2-dx^4
	14	ad 26r	$-b+cx^2-dx^4$
	15	mh 1t	$-bx+cx^3-dx^5$
	16	mh 1t	$-bx^2+cx^4-dx^6$
	17	sr* 1	$(-bx^2+cx^4-dx^6)2^{-1}$
	18	ad 27r	$(a-bx^2+cx^4-dx^6)2^{-1}$
	19	mh 1t	$(ax-bx^3+cx^5dx^7)2^{-1}$
	20	sl 1	$ax-bx^3+cx^5-dx^7$
	21	sp (0)	Return to main program
	22	0.77777	$1-2^{-15}$
	23	0.00001	2^{-15}
	24	1.77560	$-d$
	25	0.05055	c
	26	1.26521	$-b$
	27	0.62210	$a \times 2^{-1}$